Resource Efficient Machine Learning Techniques for Monitoring Repetitive Activities through Wearable Devices in Real-time

by

Yousef Kowsar
https://orcid.org/0000-0001-7606-0171

Submitted in total fulfillment of the requirements of the degree of Doctor of Philosophy

at the

School of Computing and Information Systems

The University of Melbourne

March 2020

Abstract

Weight training activities have become an inseparable part of an athlete's training to reach their maximum performance. However, at the same time, weight training is considered as the fifth costliest sport in terms of injuries [1]. Studies have shown that monitoring weight training performance is one of the most effective ways of reducing injuries caused by this type of exercise. Given the complexity of weight training exercises, it is a challenge for trainees to know whether they are performing their exercises correctly or not. Considering the fact that incorrect performance of a weight training exercise can result in life-long injuries, which may cost athletes their professional careers, it is of utmost importance to design systems that can detect incorrect performance of weight training activities.

In this thesis, we motivate the importance of personalised monitoring of weight training performance using wearable devices. We show why current supervised approaches for monitoring weight training routines fail to address the needs of professional athletes. We discuss the emerging need for resource efficient machine learning techniques to monitor weight training activities in real-time, using a wearable device. We then present a novel workflow to detect weight training performance anomalies from observing only the correct performance of an exercise by the trainee. Our workflow motivates two fundamental questions to be addressed in the time series domain: (1) Identifying a trainee's weight training performance from the incoming stream of data generated from wearable devices efficiently and in real-time, (2) Analysing a trainee's weight training performance efficiently.

We address the first question of identifying the trainee's weight training performance using wearable devices by formally defining weight training activities as intervals of recurrence—short bursts of consecutive repeating signals—from the incoming time series data. We present an efficient, online, one-pass and real-time algorithm for finding and tracking intervals of recurrence in a time series data stream. We provide a detailed

theoretical analysis of the behaviour of any interval of recurrence, and derive fundamental properties that can be used on real world data streams. We demonstrate the robustness of our method to variations in repetitions of the same pattern adjacent to each other.

We then advance our signal processing approach to monitoring weight training exercises by addressing the shape analogy of signals. *Shape* analogy is a technique where signals in the form of time series waveforms are compared in terms of how much they *look alike*. This concept has been applied for many years in geometry. Notably, none of the current techniques describe a time series as a geometric *curve* that is expressed by its relative location and form in space. To fill this gap, we introduce *Shape-Sphere*, a vector space where time series are presented as points on the surface of a sphere. We prove a pseudo-metric property for distances in the Shape-Sphere. We show how to describe the average shape of a time series set using the pseudo-metric property of the Shape-Sphere by deriving a centroid from the set. We demonstrate the effectiveness of the pseudo-metric property and its centroid in capturing the *shape* of a time series set, using two important machine learning techniques, namely: Nearest Centroid Classifiers and K-Means clustering, through 48 publicly available data sets. Our results show that Shape-Sphere significantly improves the efficiency of both techniques. Shape-Sphere improves the nearest centroid classification results when shape is the differentiating feature, while keeping the quality of clustering equivalent to current state-of-the-art techniques.

We subsequently design and develop LiftSmart: a novel smart wearable to detect, track and analyse weight training activities. LiftSmart is the first wearable for weight training that is based on unsupervised machine learning techniques designed in this thesis to eliminate reliance on labelled data. We developed LiftSmart with the ultimate goal of personalised monitoring of professional weight trainers. LiftSmart is tailored to the needs of individual professional weight trainers to monitor their performance by automatically adapting the standard performance of an exercise, which is set for each individual athlete.

In summary, in this thesis we design resource efficient machine learning techniques for monitoring weight training activities in real-time using a wearable device. We demonstrate the effectiveness of our technique in monitoring weight training activities in real-time by designing the first wearable device that automatically detects, tracks and provides feedback about any weight training activity that an athlete performs in a gym.

Declaration

This is to certify that:

1. the thesis comprises only my original work towards the PhD,
2. due acknowledgement has been made in the text to all other material used,
3. the thesis is less than 100,000 words in length, exclusive of tables, maps, bibliographies and appendices.

Preface

The core chapters of this thesis, Chapters 3 to 6, are based on the following papers. I declare that for each paper I am the main author who has contributed over 50%. Chapters 3 to 6 are respectively based on the articles A, C, D, and B.

Published papers

A) **Kowsar, Y.**, Moshtaghi, M., Velloso, E., Kulik, L., Leckie, C. (2016, November). Detecting unseen anomalies in weight training exercises. In Proceedings of the 28th Australian Conference on Computer-Human Interaction (pp. 517-526).

B) **Kowsar, Y.**, Velloso, E., Kulik, L., Leckie, C. (2019, September). LiftSmart: a monitoring and warning wearable for weight trainers. In Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers (pp. 298-301).

In revision following peer review:

C) **Kowsar, Y.**, Moshtaghi, M., Velloso, E., Kulik, L., Leckie, C. An Online Unsupervised Dynamic Window Method to Track Repeating Patterns from Sensor Data. IEEE Transactions on Cybernetics.

D) **Kowsar, Y.**, Velloso, E., Kulik, L., Bezdek, J. C., Leckie, C. (2019, September). ShapeSphere: A metric space for analysing time series by their shape. Pattern Recognition.

Acknowledgement

As my PhD journey comes to an end, there are many people who I need to acknowledge who have made this journey possible and helped me along the way.

I am very grateful to my supervisors, Professor Chris Leckie, Professor Lars Kulik, and Doctor Eduardo Velloso for their continuous support during my Ph.D. Thank you for your time and guidance to undertake my studies. Your professional and personal advice will always be with me and always guide me in my future endeavours.

I would like to thank Professor Frank Vetere and the IDL lab (formerly SocialNUI) for partially funding my project and providing me with the necessary resources to undertake my studies.

I would like to thank my advisory chair, Professor Tony Wirth for his insightful comments and taking the time to provide me feedback.

I would like to thank Doctor Massud Moshtaghi for encouraging me to turn my passion in sports and machine learning into my Ph.D. project and supporting me to define my research. I would also like to thank Professor James C. Bezdek for his research collaboration and professional advice during my studies. I would like to acknowledge Mr. Gabriele Marini for his help in developing the LiftSmart android application in my PhD.

I would like to thank my parents and my in-laws for their continuous support to me and my wife during my Ph.D. Their support was a comfort in the time we needed most.

I am deeply grateful to my wife, Mahtab Mirmomeni, for her continuous support and encouragement during my Ph.D. I am grateful for all her patience and support in challenging times and her positive attitude and thoughts that kept me going through my Ph.D.

I would also like to thank my son Kiyan whose presence always brings me smiles and hope to my heart. I started my PhD when he was 3 months old and he has been a delight when our family needed it most.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Wearable sensors that can monitor and analyse the activity of an individual have the potential to provide real-time insights about an individual's health. For example, wearable sensors can be used to monitor athletes while performing exercises. This monitoring can provide feedback to athletes and their trainers [3] to help prevent injuries, which currently cost over a billion dollars to treat each year worldwide [4]. An open challenge is how to extract such feedback from the data provided by wearable sensors. These sensors generate streams of motion/biomedical data in a time series form, which can be generated by sampling an underlying continuous signal, such as the movement patterns of an athlete while performing an exercise.

Analysing the captured data in real-time on wearable devices plays a key role in developing warning systems that can detect abnormal activities as they occur, which is crucial in applications such as detecting abnormal performance by athletes to prevent injuries. The challenge with designing methods that can run on a wearable device is that they must be sufficiently efficient to run on a device with limited computing resources [5].

The current state-of-the-art for detecting and tracking the repeating movement pattern on a wearable device is based on supervised methods, where a classifier is trained using labelled data [6, 7, 8]. However, labelled data is expensive to collect and the training

itself is also computationally intensive, requiring the tuning of multiple hyper parameters [9]. Moreover, supervised methods fall short in detecting unseen repeating patterns or variations of the same repeating pattern. Thus, we require unsupervised methods that can adapt to personalised routines and unseen exercises without the need for extensive labelled training data. Moreover, any model that is used to learn the movement patterns of repetitive exercises must also be robust to environmental noise, such as slight variations in normal movement, variations in the positions on the body of the wearable devices, and noise in the readings of the wearable device.

In summary, a successful machine learning approach that can run on a wearable device to provide meaningful feedback to its user about their repetitive exercises must satisfy the requirements in Table 1.1. These requirements are the basis for the design goals of such a monitoring system, and these goals motivate the research questions that we address in this thesis.

Table 1.1: Our design goals and their impact on the system for monitoring repetitive movement patterns in weight training exercises. The Chapter column indicates the corresponding chapter in the thesis that addresses each design goal.

G#	Design Goal	Impact	Chapter
1	**Detects and tracks in real-time**	Real-time monitoring	4
2	**Analyse in real-time**	Real-time feedback	5
3	**Runs on wearable devices**	Ubiquitous usage	6
4	**Performs analysis unsupervised**	Adapt to new/unseen routines	3
5	**Adapts to variations**	Personalised	4
6	**Handles environmental noise**	Robust to environmental noise	4

Next, we discuss our design goals for the weight training application: the real-life application that derives from our research using two real-life weight training scenarios. We elaborate on the challenges that arise from our design goals. We define the problem of monitoring weight training activities in real-time using a wearable device in Section 1.2. In Section 1.3 we provide an overview of this thesis in terms of how each chapter is designed to address our design goals.

1.1 Monitoring weightlifting moves in real-time

Free weight training is the process of building muscles using weights such as dumbbells, barbells, kettlebells and sand bags. It has become a common form of strength training that has been advocated globally by medical authorities. For example, at least two sessions of 45 minute strength training per week is recommended by the American Heart Association [10], the Australian Government Department of Health [11] and the World Health Organisation (WHO) [12]. In free weight training, a session consists of a few (often six to ten) sets of different exercises. Each exercise is designed by an expert and focuses on strengthening a single or a group of muscles through muscular contraction. A weight training exercise is a particular movement that makes muscles work against the weight, which results in strengthening the muscle. Each set consists of the continuous and uninterrupted performance of a repetitive exercise for at least three repetitions. Repetitions in a set are performed consecutively (with no rest in between them). The trainee rests after each set to allow the activated muscle(s) to recover before starting the next set. The goal in strength training is to increase the intensity of each set gradually (increasing the weight and/or the number of repeats), which in turns results in increasing the strength capacity of the muscle group involved in the exercise. However, Drew and Finch [13] showed that strength training injuries are highly correlated to increasing the intensity of the sets, even in expert trainees. Gabbett [14] showed that incorrect posture and inappropriate load are the two main reasons behind most injuries. Therefore, the consensus among experts in weight training is that it is of the utmost important to monitor athletes' performance during their training to prevent injuries [15, 16, 17].

1.1.1 A trainee-trainer scenario

Monitoring a trainee is an important strategy to ensure that the trainee performs the exercises correctly. At present, this monitoring is performed by the trainers. This trainer-

trainee interaction usually follows a cyclical loop [18]. First, the trainer designs a program for the trainee based on the trainee's **personal** needs. The trainee performs the routine for several sessions under the trainer's supervision. The trainer identifies further strengths and weaknesses in the trainee's performance, which are then taken into consideration to design the next exercise routine for the trainee. The practice is to increase the intensity of the exercises gradually under the trainer's supervision to achieve the trainee's weight training goals.

However, increasing training loads are a major cause of a large portion of soft tissue injuries, as shown by Gebbett [14]. These changes result in muscle fatigue, which shows itself through the following signals [19]:

1. Decreasing the maximum performance that is observed;
2. Decreasing load tolerance;
3. Extending the recovery time needed;
4. Disrupting the correct posture for the exercise by the trainee.

Thus, a real-time monitoring platform that can warn a trainee **as soon as** the early signs of muscle fatigue appear is an essential step in helping both elite athletes and beginners to avoid injuries. This platform is currently implemented manually through the personal trainers' supervision.

In the following two scenarios, we discuss why it is essential for both novice and elite trainees to access a monitoring platform during their weight training routines.

> *Mark joins a gym for the first time. At the gym, he meets Jane, who was assigned as his personal trainer for his first session. Knowing that Mark's goals include increasing his overall strength, Jane designs a 6-week program including six different free-weight exercises, each to be performed in three sets of ten repetitions. She demonstrates the correct execution of each exercise and after each demonstration she asks Mark to do a few repetitions to ensure that he understood it. She gives*

him feedback to improve his technique until reaching an acceptable performance level. After going over all the exercises, Mark feels confident that he understood them. However, next time Mark comes to the gym, without Jane's supervision, he is not quite sure whether his performance is correct.

This scenario demonstrates how essential it is to monitor novice trainees. An ***in-place*** feedback system can alert both the trainee and their trainers about incorrect performance of a routine, and helps guide beginners to ensure they are on track to achieve their healthy lifestyle.

The next scenario illustrates how an expert trainee, without adequate monitoring, is prone to mistakes.

Ronnie is an experienced weight lifter. He is constantly trying to push his limits at every gym session, progressively increasing the weights in his exercises. Though he demonstrates complete mastery of the technique using light weights, the heavier the weight, the more difficult it is to maintain a proper technique. The physical and cognitive overload of the heavy weights makes it very difficult to perform the full range of motion while performing a deadlift. As a result, he gradually bent his spine and he ended up straining his back. This scenario shows that depending on the weight being lifted, even an experienced lifter, who has previously shown correct performance on a given exercise, can make mistakes in subsequent exercises.

This scenario illustrates the necessity of an ***in-place*** and ***real-time*** monitoring system that can be used by elite athletes to avoid unwanted injuries and help them stay focused in achieving their goals.

Research has identified many factors to monitor during an athlete's performance of an exercise, which can be categorised as follows [19]

1. Internal—related to biological measurements such as heart rate, blood pressure or oxygen consumption.

2. External—objective measurements of the work performed, such as acceleration, distance or power output.

Although internal measurements are an accurate indicator of the overall health of the athlete and how they cope with the workload, they cannot provide any insights into the accuracy of the athlete's posture. This feedback to an athlete about their posture has been shown to have tremendous impact on both learning a task and excelling in the task [20].

A common approach for monitoring an individual is through the use of a personal trainer (PT) where a PT monitors the trainee while performing an exercise to ensure they adhere to correct technique throughout the workout. However, accessing a PT is not a feasible long term solution. First, hiring a PT is expensive, e.g., between $50 to $100 per session. Thus, even adhering to minimum recommendations for strength training can cost up to $1000 per month in addition to the cost of gym membership and extra equipment. Second, working out with a PT means that trainees must arrange their time around their PT, which can be a nuisance. Third, working out with a PT often means binding the trainee to work out in a specific gym or place, which restricts the trainee's freedom to perform their exercise wherever they want, whenever they want.

The ubiquity of motion sensors makes them an appealing solution to provide automated feedback on a user's free weight exercise technique. Users can access motion sensors from off-the-shelf utilities, such as phones, headphones and smart watches. This accessibility has increased the capacity to ***continuously*** and ***ubiquitously*** collect data from individuals while performing an exercise. The collected data can be used to provide in-place and real-time feedback to the trainees and thus address the scenarios described above.

1.1.2 Challenges

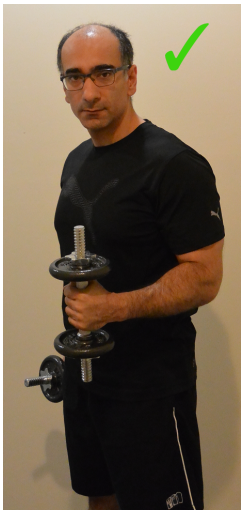
A major challenge in automatically monitoring weight lifting exercises arises from the human body's large number of degrees-of-freedom in movement. The large number of degrees-of-freedom in movement creates a huge potential for possible incorrect postures for any given exercise. In addition, these degrees-of-freedom result in multiple variations of any exercise that targets specific muscle fibres, where these variations do not necessarily represent a mistake. For example, the exercise in Figure 1-1 could also be performed by twisting the dumbbell while lifting. Any monitoring platform must be able to consider these variations and be able to differentiate them from an incorrect performance of the exercise. A successful system must be robust to ***variations of patterns*** so that trainees can achieve their personal goals.

Another challenge arises from the nature of weightlifting exercises. In weightlifting, sets are often performed in three-to-many repetitions of a single exercise, with each repetition being the unit of analysis. Each repeat takes at most ten seconds. Each set lasts at most one minute, which is surrounded by a rest period of a few minutes duration where the trainee is free to perform any activity, such as sitting on a bench resting, going for a walk or drinking some water. Thus, a successful method should be ***robust to environmental noise*** and able to differentiate the weight training intervals from other activities that a trainee might perform at a gym.

In summary, any online and real time monitoring system for weight lifting must be robust to variations of an exercise, while being efficient enough to identify when the trainee starts performing an exercise, as well as efficiently detecting and qualifying each repetition of an exercise in the set.

(a) Biceps curls starting position

(b) Biceps curls movement



(c) Correct variation

(d) Incorrect movement

Figure 1-1: Unilateral biceps curl performance. Top pictures show correct performance. Bottom row shows two variations. Left is a correct variation (with a green tick) and right is an incorrect variation (with a red cross).

1.2 Problem statement

To illustrate the characteristics of weightlifting data, Figure 1-2 illustrates a sample data stream recorded for three different weight training exercises, performed by a trainee using a motion sensor attached to his wrist. The data is recorded in quaternion units that show rotation in space. Each exercise set is delineated from the rest of the data stream using vertical bars. The repeated pattern, which shows the exercise performed by the trainee, is shown in red for each set. Comparing each exercise set, Figure 4-1 shows how much each

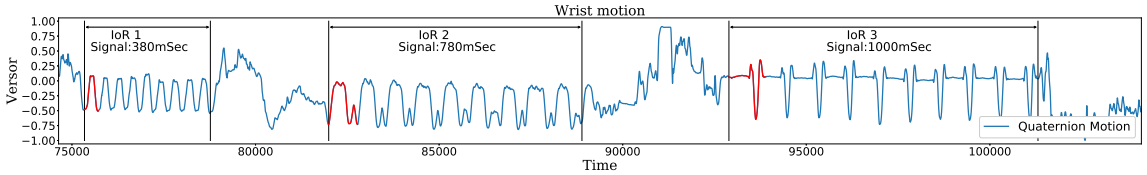


Figure 1-2: A recorded sample of motion data from three weight lifting exercises. The three sets are shown with the recurring exercise in red in each set.

repetition (red segments from each exercise interval) changes from one set to another in both shape and frequency. We see that the shape of the data not only varies from one set to the next (as indicated by the numbered regions), but also from one repetition to the next (with different levels of amplitude). We can also see how the data corresponding to the rest periods is substantially different to the data during the exercises.

Our focus in this research is to design machine learning methods that can help people with knowledge of weight training. The common characteristics of this focus group is that they all know how to perform an exercise. Thus, they can perform the first few repeats of their exercise with no problems. However, as they progress towards the end of their set they can face muscle fatigue, which can cause trainees to deviate from their routine to finish their set. Our goal in our research is to design machine learning techniques that can detect the personally correct performance of an exercise and then track each performance of the exercise according to the trainees' personal best. Examples of such deviations can include a smaller range of motion while progressing through an exercise set, or the existence of fundamental coloured noise in the signal recorded from the set that gradually changes the shape or frequency of the detected signal after a few repetitions of the exercise.

1.3 Contributions

In this thesis, our focus is on designing resource efcient machine learning techniques that can run on wearable devices to analyse human movement in-real time and in-place.

We focus on the limitations of current machine learning techniques in this context, in terms of achieving the design goals for such a wearable device as listed in Table 1.1. In Chapter 2 we describe in detail the currently available systems for tracking and evaluating weightlifting exercises (Section 2.1). We examine each system's underlying machine learning techniques and their limitations in terms of addressing our design goals in Table 1.1, i.e., robust, unsupervised learning for real-time, personalised monitoring. Therefore, our goal in this thesis is to design machine learning techniques that address these machine learning challenges. We showcase the usability of our techniques through a proof-of-concept wearable that can detect, track and evaluate weightlifting movements in real-time and in-place. We showcase the potential our machine learning techniques create for future research in Human Computer Interaction (HCI), which enables warning systems for weightlifting movements in this context. However, further HCI research on this proof-of-concept is outside the scope of this thesis and is left for future research.

In Chapter 2 we overview the current state of the art techniques for monitoring weight training activities. Since our machine learning techniques are motivated through the weight training monitoring application, we first discuss the state-of-the-art systems that address our motivation. We go into details of how each method performs the monitoring task and discuss its pros and cons in regards to satisfying our design goals discussed in Table 1.1. We then review the machine learning techniques that address our analytical challenges and discuss how each of the current methods fails to satisfy all our design goals discussed in Table 1.1.

In Chapter 3 we present a novel workflow to detect performance anomalies of weight training activities from observing only the correct performance of an exercise by the trainee. We evaluate our method on a benchmark data set for the biceps curl exercise and evaluate our system with a publicly available dataset, and show that our workflow detects unseen incorrect weight lifting performance with 98 percent accuracy. The result of this chapter is published in paper A in the Preface.

In Chapter 4 we formally define weight training activities in terms a burst of consecutive recurring signals from a time series data stream which we called an Interval of Recurrence. We present an efficient, online, one-pass and real-time algorithm for finding and tracking intervals of recurrence in a time series data stream. We provide a detailed theoretical analysis of the behaviour of any interval of recurrence, and derive fundamental properties that can be used on real world data streams. We show why our method, unlike current state-of-the-art techniques, is robust to variations in repeats of the same pattern adjacent to each other. We also discuss how the applications of the interval of recurrence goes beyond weight training activities and can be seen in various fields of research, such as the energy consumption of air conditioning in data centres can be modelled as intervals of recurrence, or in seismology where different stages of earthquakes can be modelled by intervals of recurrence. We test our algorithm on real-world weight training datasets, which showed that our method can detect intervals of repeating activities on wearable device with high accuracy (over 70% F1-Score) and in a real time environment with only a 1.5-second lag. Our experimental results from real world datasets demonstrate that our approach outperforms state-of-the-art algorithms in both accuracy and robustness to variations of the signal of recurrence. The results of this chapter are under review as paper C in the Preface.

In Chapter 5 we continue modelling weight lifting exercises using signal processing theories from Chapter 4. We model repetitions of an exercise using signals in the form of time series data. Thus weight lifting exercises and repetitions can be compared through their signal properties. We investigate the *Shape* analogy technique in geometry that concerns comparing geometric objects by their shape. We introduce this geometric '*shape*' analogy into time series analysis by describing time series data as a geometric *curve* that is expressed by its relative location and form in space. We then transform the computed curve into a vector space where each time series is presented as points on the surface of a sphere (Shape-Sphere). We prove a pseudo-metric property of shape

distances in Shape-Sphere and show how to compute the average shape in this space. We demonstrate the effectiveness of these two properties in analysing time series data by applying them into a Nearest Centroid Classifier and K-Means clustering, through 48 publicly available data sets. Our results show Shape-Sphere improves the classification results when shape is the differentiating feature, while keeping the quality of clustering equivalent to current state-of-the-art techniques. The results of this chapter are under review as paper D in the Preface.

In Chapter 6 The outcome of this chapter is to build a proof-of-concept wearable that can be used in the future of sport science. The wearable can be tuned to detect and evaluate different exercises based on their recorded signal. However, the correct levels of thresholds for warning a trainee requires interdisciplinary research between sport science, physiology and machine learning that goes beyond the scope of this thesis and has been left as future research. In this chapter we present LiftSmart, a novel smart wearable to detect, track and analyse weight training activities. LiftSmart is the first wearable for monitoring weight training activities that is based on unsupervised machine learning techniques to eliminate the use of labelled data, which is expensive to collect, computationally intensive, and requires the tuning of multiple key parameters. LiftSmart, uses the workflow from Chapter 3 to monitor weight training activities. LiftSmart accomplish its monitoring goals using the techniques in Chapter 4 and 5 to detect, track and analyse weight training activities in real-time, on the wearable device. Thus, it is the first wearable that monitors (i.e., detects, tracks and evaluates) weight training activities without the use of labelled data online and in real-time. The results of this chapter have been published as paper B in the Preface.

In Chapter 7 we elaborate the final conclusions of this thesis. We discuss what has been achieved in this thesis and leave a research blueprint that can be regarded as a starting point for future researchers.

Chapter 2

Weight training activity monitoring methods: background review

As we discussed in Chapter 1, the main focus of this thesis is to design machine learning techniques that can address the shortcomings of current learning techniques to fulfil our design goals mentioned in Table 1.1. As such, in this chapter, we review both current weight training systems that are designed to detect, track and evaluate weight training movements. We discuss the machine learning techniques used in these systems, and discuss why none of the current machine learning techniques is capable of addressing our design goals. After reviewing the available systems for tracking weight training activities, we continue by reviewing the available machine learning techniques that have been used in the literature to design the weightlifting systems in more detail. We discuss in detail the shortcomings of these techniques in answering our design goals.

2.1 Monitoring settings: How to monitor trainees?

Monitoring settings define the tools and the environment that are used to monitor trainees. It comprises the types and amount of data available for analysis, the computing resources that can be used, as well as the place where trainees' performance is monitored.

The monitoring setting can be divided into two categories: (1) Instrumented settings, and (2) Ubiquitous settings.

2.1.1 Instrumented settings

An Instrumented setting is a physical area where monitoring tools are installed. Trainees must perform their routines in this area to be able to receive their performance feedback. Since the space is dedicated to performance monitoring, it can have access to sufficent computing facilities to analyse the trainees' performance. As a result, computationally intensive machine learning techniques can be routinely applied to analyse trainees' performance. Common technologies that have been used in instrumented spaces are: digital photo/video cameras, augmented reality cameras and motion-sensing devices such as the Microsoft Kinect (`https://azure.microsoft.com/en-us/services/kinect-dk/`).

Method: Capturing the trainee's body image (human figure) is the main approach used in instrumented spaces that are dedicated to exercise monitoring. Video or still-image cameras are used to capture the body figure of the trainees to detect and track their activities. The main challenge in these methods is to accurately identifying the trainee's figure in an image, and then follow it in the future frames to track their performance. The use of photo cameras was made possible by Israd and Blake [21] who designed the first method to extract a continuous curve around an object from the recorded video and track the curve in the future frames. Rittscher et al. [22] used this contour tracking approach for detecting activities such as jumping, star-jumping and ball-throwing activities. Other approaches have been studied, such as movement capture from an object's mesh extracted from a video recording [23, 24, 25], or capturing joint movements from multiple camera angles[26, 27]. Deep neural networks have provided more accurate methods to detect multiple rigid bodies from video, which allows more advance systems to be used in practice for detecting and tracking exercise activities. Adversarial neural networks

for pose estimation of multiple people in a crowd [28], convolutional neural networks for tracking multiple people's pose in a crowd [29], and statistical inference to associate body parts to their related rigid body in real time regardless of the number of people in the video frame [30], are examples of pattern recognition techniques that allow the use of video capture devices to detect and track multiple trainees in a gym. Khurana et al.[31] introduced the first system (GymCam—2018) that successfully combined neural networks and rigid object trajectory extraction to detect, recognise and track weight training activities in a gym. They used a traditional trajectory and movement tracking method [32] to detect movement from a gym scene, and clustered the trajectories into different activities. This data was then analysed by a multi-layer perceptron to accurately identify and track each movement.

As mentioned, the main challenge for video analysis is to detect and track multiple rigid bodies in a captured video. In another approach to solve this problem, Microsoft introduced the Kinect to automatically detect and track rigid bodies in front of its motion-sensing camera. This extraction of a rigid body makes Kinect a more approachable solution for designing weight training activity trackers. For example, one of the earliest use cases of the Kinect was MotionMA [18], which was designed to guide a trainee while they performed an exercise in front of a Kinect camera. The system learns the correct performance from an expert by capturing the skeletal figure of the expert, and then guides the user through the exercise by showing them how to move different parts of their body. In another approach, YouMove [33] was designed with the goal of learning moves by breaking down each move into simple components. Zhao et al. [34] used the Kinect to automatically evaluate physiotherapy exercises performed by patients in the home.

Strengths and Shortcomings: The main strength of instrumented spaces comes from the quantity and quality of data available for analysis. These areas are pre-designed for particular exercises. As a result, cameras can be installed to capture the most informative

views of trainees' performance. This particular factor makes instrumented spaces an ideal environment for elite professional athletes, where cost is less of an issue.

The main challenge in instrumented spaces is that analysing their data requires powerful computing resources, which can be considered as invasive to the user's privacy. In these techniques, the analysis is based on capturing the full figure/silhouette of the trainee, which requires substantial computing facilities for analysis and thus can reveal unwanted/unnecessary information. Although these techniques are popular with elite/Olympic athletes, they are limited to situations where users sign consent forms and are willing to share this information because of the performance benefits they can receive and the trust they have in the system.

The instrumented space fails to address the ubiquity requirement of our design goals. This limitation stops the user from being able to exercise wherever they want. For example, the ability to monitor the same person after switching from one gym to another or exercising by the beach is limited with the use of these technologies.

2.1.2 Ubiquitous settings

To relax the need to restrict the trainees' monitoring to a particular location, ubiquitous systems have been designed. The unique characteristic of these systems is their portability, i.e., trainees can easily move or wear them in a range of locations. In order to achieve the portability design goal for these methods, they must address the challenge of being able to extrapolate movements from limited data that does not capture the entire trainee's body figure. Inference from the impact force on a surface and inertial measurements are two common methods that have been studied in this category of systems.

Method: *Impact of force* is based on the idea that trainees can use some specific apparatus such as standing on a mat or wearing a glove, which can sense the amount of force applied to its surface. Examples include standing on a mat or holding a specific piece of

equipment like a chin up bar. The equipment then provides feedback to the trainee by analysing the forces that can be captured from its surface. SmartMat [35] was designed based on this idea through capturing the pressure from a user's body on the mat's surface during exercise. The mat identifies 10 pre-trained exercises by analysing the pressure it receives from the trainees body. In a closely related approach, Smart-Surface [36] classifies Yoga movements by using surface pressure sensors. Zhou et al. [37] designed a fabric as a wearable to detect leg exercises by observing leg muscle activities through surface pressure. In this approach, repetitive activation of the leg muscle is considered an exercise. Other related activity tracking approaches are the use of textile deformation [38] to identify movements from texture deformation. Alternatively, textile movement from a glove [39] senses pressure on a glove to detect and analyse indoor activities.

Inertial measurement is based on local features extracted from a trainee's limb movements during an exercise. Inertial Measurement Units (IMU) are the main type of sensor that has been used for capturing these features. IMU sensors are popular because of their ability to observe the movement without depending on any external sources such as cameras, and thus can operate in a non-invasive way. These methods use only limited movement measurements rather than external imaging and thus provide both greater privacy as well as a ubiquitous systems that are accessible to a wider range of users in a variety of settings. These sensors generate streams of motion data in a time series form. These time series are generated by sampling the underlying continuous movement as a waveform, such as the arm movement of a trainee while performing an exercise. For example, a gyroscope can sample the angular rotation of the limb it is attached to in 3-directions, which can be used to monitor arm or leg exercises.

One of the earliest use cases of IMU sensors for weight training was by Chang et al. who tried to detect weight training exercises [40]. They achieved 80% accuracy in detecting what a trainee has performed during a weight training scenario, and were able to count the number of repetitions by training a Support Vector Machine (SVM). Many

studies have applied the same techniques to detect and count the number of repetitions of an exercise. RecoFit tried to detect multiple weight training exercises using a wearable sensor on the trainee's wrist [6], using an extensive set of statistical features extracted from the recorded time series. They trained a Support Vector Machine (SVM) that achieved over 90% accuracy for 13 predefined exercises. In another approach, NuActiv was designed to answer the problem of finding unseen weight training activities [41] through semantic attributes of predefined exercises. In recent years there have been many attempts in improving detection and counting accuracy of exercise repetitions in weight training scenarios. Pruthi et al. [7] designed Maxxyt, a heuristic sensor fusion approach to extract unknown exercises from recorded sensor data. They reported a "within 2 repetition" accuracy of 98% for 10 different exercises in an offline scenario. Shen et al. [8] designed MilFit, a smart watch that can classify weight training activities from other exercises such as running (aerobic activities), and count the number of repetitions from the weight training exercises in an offline mode. Advances in deep neural networks opened the opportunity to achieve higher accuracy. Um et al. [42] used a convolutional neural network (CNN) to classify 50 different weight training activities. They reported 92.1% accuracy in classifying the exercises. Skawinski et al. [43] used a CNN to count the repetitions of an exercises, which reported 97.9% accuracy in counting the repetitions of an exercise set.

Strengths and weaknesses The main disadvantage of ubiquitous systems comes from their compact design. These devices are designed to be portable, and thus rarely have access to powerful computing resources. As a result, resource efficient machine learning techniques are needed that can run on these devices.

However, the small size of ubiquitous systems gives them a portability and affordability advantage over instrumented spaces. Wearable watches are now off-the-shelf devices, which can be purchased from less than one hundred dollars. Trainees can easily buy and

wear these devices at a wide variety of times and locations. The ubiquity and affordability of these devices not only make them a flexible tool to monitor trainee's everyday activities, such as running by the beach or training at the park, but also give them the advantage of being able to provide a low-cost warning system for a wide variety of trainees.

Ubiquitous methods are also popular because of their ability to observe the movement of a trainee without depending on any external sources such as cameras, in a non-invasive way. This not only reduces cost and ease-of-use, but also helps protect the privacy of the trainee through the use of more limited sensing, in contrast to video-based approaches that capture detailed images of the trainee.

2.2 Approaches to analysing trainees' performance

Our aim in monitoring a trainee is to check whether they are performing their exercise correctly. This process involves observing and recording a trainee's activity and then checking whether they have performed the exercise to a desired standard. As discussed in Section 1.2, this process starts by detecting the exact time a trainee starts performing an exercise, based on the observed data. Then, each repetition of the exercise needs to be identified and tracked throughout the exercise set from the observed data. The final step is to evaluate each repetition. Machine learning techniques to perform detection, identification, tracking and evaluation of exercises can be categorised using the type of knowledge they require to analyse the data:

1. Posteriori knowledge: knowledge that is derived from experience after the analysis of data collected in a controlled environment, and
2. A priori knowledge: knowledge that is based on general principles of the analysis task, rather than being extracted from empirical analysis of training data.

We discuss the trainees' monitoring steps according to these two categories of knowledge.

2.2.1 Monitoring from pre-recordings: A posteriori approach

Machine learning techniques that fall into this category are based on data sets that are pre-recorded in a laboratory/simulated environment. In these settings, researchers acquire the required data in a controlled environment. They first select a set of activities to monitor. The next step is to recruit people who perform the activities in the controlled environment. During this step, researchers collect the same type of data that is available from the final product. The next step is to label the collected data. In this step researchers and field experts must work together to label the data, where the data is analysed by the expert to (1) label the segmentation of the exercise set, i.e., identify that an exercise is happening; (2) label each exercise performance within a set; i.e., segment the detected exercise set into repetitions; and (3) evaluate each repetition; i.e., evaluate the accuracy, intensity, etc of the repetitions.

Method: One of the earliest studies of weight training activity detection and recognition was by Chang et al. who tried to detect weight training exercises [40]. They selected nine different weight training exercises to monitor. For data collection they used eight male and two female subjects who each perform the nine exercises in a laboratory. Each subject performs each exercise for three sets of fifteen repetitions. Their subjects were allowed to perform more repetitions if they want to or stop if they feel tired or pressured. The researchers manually segmented the data by starting and stopping the recording of data when subjects were performing the exercises. In summary, they were able to collect 4925 repetitions, which were performed over 162 minutes. They manually count the repetitions within each set and used this as the label of that set, together with the exercise type. They used a Naive Bayesian Classifier to classify the data. They achieved 90% accuracy in detecting what a trainee has performed during a weight training scenario. They were able to count the number of repetitions using a Hidden Markov Model (HMM), which achieved an average 5% miss-count with a maximum miss-count of 17%.

In another study, Morris et al. [6] presented RecoFit, a wearable tuned for weight training exercises. They collected a dataset from 94 participants in a laboratory environment to train an SVM. They use expert personal trainers to watch the participants while they perform the exercises. The trainers label the dataset in terms of detecting the start and end of each set, identifying each repetition and counting them. They collected 126 sessions of data before being able to train a SVM. They used an extensive set of statistical features extracted from the recorded IMU data to train the SVM, which achieved over 90% accuracy for 13 predefined exercises.

In another approach, NuActiv was designed based on the idea of classifying activities using simple limb movement features. In this method an activity is defined by a vector of simple limb movements [41]. The idea is that by learning this set of features, NuActiv can detect complex activities through these features. They selected seven features to learn by collecting data from 20 subjects. They showed that their system can achieve 70% accuracy in detecting 10 different exercises. However, they reported that the system can tend to be confused between similar exercises that differ in only one feature.

Strength and weaknesses The main strength of posteriori approaches is that they learn a specific exercise in detail. As a result, these systems are useful to monitor a set of predefined exercises that are often given to new gym trainees. In this scenario, novice trainees are given a set of certain exercises that help them learn wight training, build necessary strength and flexibility.

However, after a few months of training, trainees require more personalised routines that are tuned for their personalised needs and goals. Thus, trainers change the standard routine for their trainees, i.e., variations of the same exercises, more compound exercises or new specific routines are designed for trainees. As discussed and shown by different studies, the existing pre-trained classifiers fail to account for these personalised changes [40, 44].

Studying different weight training databases reveals that even a simple database can include over a thousand exercises (e.g., bodybuilding.com has 1000 exercises, jefit.com has 1309 exercises, and ExRx.net has 1800 exercises). Adding the variations of each exercise to this list together with personal variations means that it is both costly and time consuming to collect, label and learn every exercise (if possible at all in a timely manner). Further, it has been shown that supervised template boundaries for weight training are not easily generalisable to thousands of exercises [45]. To the best of our knowledge there is no research showing a supervised classifier can detect hundreds of weight training exercises. Thus, methods based on posterior knowledge fall short in detecting and tracking all possible exercises for a trainee.

2.2.2 Monitoring using weight training insights: A priori approach

An a priori approach is a method that rests on an insight from the problem domain. For example, in weight training we might consider any repetitive movement to be an exercise routine. However, this intuition fails in certain situations, hence such a naive approach will fail. Nevertheless, if analysis can be applied to data that is assured to be received from a weight training routine, then many of the posteriori challenges can be solved, such as limited availability of labelled data and personalisation of exercises. To the best of our knowledge, no method has been designed to detect, track and analyse weight training routines using purely a priori knowledge. Thus, in this section we present methods that can be used to perform each task without requiring the use of labelled data. Since our focus is on approaches to analyse time series data received from IMU sensors, we focus on these methods.

IMU sensors generate streams of motion data in a time series form. This time series is generated by sampling the underlying continuous movement as a waveform, such as arm movements while performing an exercise. For example, a gyroscope can sample the angular rotation of the limb it is attached to in three directions.

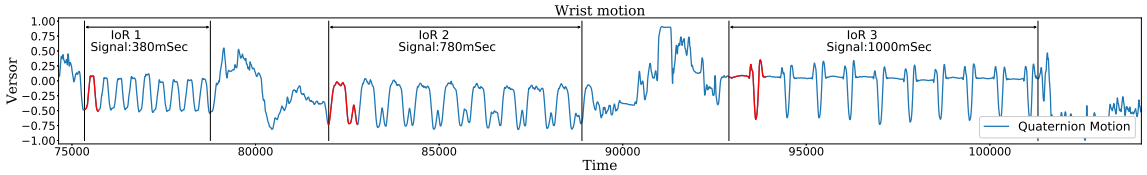


Figure 2-1: A recorded sample of motion data from three weight training exercises. The three sets are shown with the recurring exercise highlighted in red.

A time series is a set of sequential observations collected over time. The main characteristic of this set is that the value of each observation is related to its index in the set and the previous observations [46, 47]. Time series analysis is concerned with extracting these relations to infer information from the available time series within a data set. To capture these relations we first need to represent the time series, which we discuss in the next section.

In the Problem Statement in Section 1.2, we discussed that weight training exercises manifest as consecutive repeating patterns within the time series received from an IMU sensor (see Figure 2-1). This figure shows three sets of different weight training exercises. The start and end of each set is shown by vertical black lines. The waveform associated with the exercise performed in each set is shown in red. A closer look at Figure 2-1 reveals that detecting the start of an exercise is equivalent to finding the starting point of each set from the time series—a problem studied under ***time series segmentation*** in the literature. Identifying and tracking the exercise within a set is equivalent to identifying the repeating waveform inside the set. Finally, evaluating a trainee's performance is equivalent to evaluating the repeating waveform inside the set, which is studied as finding the ***time series similarity*** in the literature. Next, we discuss methods to represent time series data in the computer and then review current time series methods that address each step of the exercise monitoring workflow.

Time series representation

Capturing and analysing time series data can be an efficient method for representing the data, namely:

1. Feature representation,
2. Fragmented representation, and
3. Continuous representation

of the time series data.

Feature representation In feature representations a set of temporal features such as the mean, standard deviation or dominant frequency of a time series is extracted to represent the time series. In these methods, a time series or a sub-region of the series is modelled through a small number of aggregated features that are extracted from the original data. The goal is to represent the data in a way that feature based machine learning techniques such as Support Vector Machines (SVM) can be applied to the time series data set [48, 49]. Techniques such as statistical feature extraction [50], dominant frequencies [51, 52], signal features such as temporal energy of a signal from the time series [53], and perceptually important points (PIP) [54, 55] are a few examples of methods that have been successfully applied as feature based machine learning models for time series data by extracting features from the data.

Fragmented representation In fragmented representation, a time series is represented through its segments. A segment is an ordered subset from a time series containing consecutive observations from the time series. Yi and Faloutsos designed an approach that represents each segment through its mean value [56]. In their approach, segments are generated by segmenting the time series into equal length parts. Keogh et al. [57] generalised this method by introducing Piecewise Aggregation Approximation (PAA),

which generates variable length segments. An example of PAA for a time series is given in Figure 2-2.

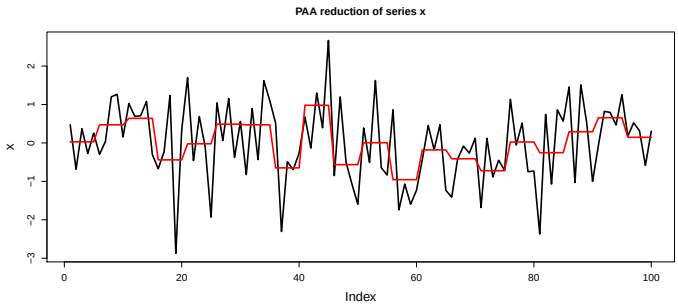


Figure 2-2: A sample time series (the black graph) and its Piecewise Aggregation Approximation representation (PAA) shown in red.

Lin et al. extended the PAA idea by introducing the statistical distribution of points in the time series to design the Symbolic Aggregation Approximation (SAX), which represents a time series using a set of predefined symbols [58]. SAX segments a time series and then maps each segment into a symbol according to the average value in the segment and the time series distribution. Piecewise linear aggregation (PLA) is another approach that represents a time series through a linear approximation of its segments [59, 49].

Continuous representation In continuous representations, the members in the time series are directly used to analyse the data. In this approach, a time series is seen as a waveform (a signal). Thus, signal processing methods, such as correlation [60, 61], Principal Component Analysis (PCA) [62, 63, 64], Fourier transform analysis [52, 65], and wavelet analysis [66, 67] are popular methods that have been used to analyse time series data as a waveform. In Section 2.2.4 on time series analogies, we discuss the approaches of analysing the time series as a waveform in more detail.

2.2.3 Detecting exercise sets within a time series—A time series segmentation problem

Lara and Labrador define the human activity recognition problem (HARP) to be the problem of identifying a temporal segment from a given time series, where one and only one activity can be assigned to the segment [68]. Thus, the activity of interest can be extracted and tracked from a time series data. Segmentation is the process of partitioning time series data into segments. The segments often associate with meaningful patterns presented in different parts of the time series [69]. Lin et al. categorised time series segmentation into three sub-classes [70]: (1) Physical boundaries, where segments are typically defined through changes of movement; (2) Derived metric boundaries, where segments are defined through a transition in variance or a threshold; and (3) Template boundaries, where segments can be defined through user defined templates. Next we review methods that are available for each sub-category.

Segmenting time series into intervals

Traditionally time series data were segmented into equal length regions [71]. This technique oversimplifies the problem of time series segmentation. Meaningful patterns often appear with different lengths in time series. Thus, an equal length segmentation can break the pattern of interest into multiple segments, which in turn can result in missing the pattern [69]. Keogh et al. [59] designed one of the first variable length segmentation algorithms. In their method a time series is segmented through piecewise linear approximation of the time series. Other methods for segmenting a time series into intervals then morphed to the problem of representing a time series by its fragments, which we discussed in Section 2.2.2.

Change Point Detection Change Point Detection (CPD) techniques are one of the earliest approaches for segmenting a time series. In CPD, the main task is to search for

properties whose changes uniquely differentiate the start and end point of the segments in the time series.

Traditionally CPD algorithms can be divided into two categories: (1) Offline algorithms, and (2) Online algorithms. Offline algorithms expect to receive the whole time series in advance before performing any processing. These algorithms can apply computing/data intensive analysis to correctly segment the time series [72, 73]. Aminikhanghahi and Cook defined both online and offline approaches under one umbrella of ϵ-real-time algorithms [74]. In their definition the ϵ value defines how "fast" the algorithm works. For example, when ϵ is equal to the length of the time series then the approach is offline, and when ϵ has a constant upper bound it is online. By their definition all the methods described in the Fragmented representation of a time series are defined as offline methods for CPD. Using this definition, they identified a statistical CPD approach as a starting point for online methods, which we discuss next.

Statistical segmentation: Statistical pattern mining is a popular method in CPD where each segment of a time series is generated from a distribution that is differentiable from its immediately previous and next segments. Popular methods in this technique are based on estimating the distribution of each segment. For example, Bayesian probability change points can be defined in terms of segments whose observations are generated from a normal distribution but with different parameters [75, 76, 77]. Gaussian processes are a generalised version of Baysian methods for change point detection [78]. Chiu et al. showed how the expectation maximisation (EM) algorithm, a process that is used in generalised statistical modelling, can be used to segment a time series [79]. One method that generalises statistical pattern mining to multiple time series is [80]. In this approach, a bundle of co-evolving time series are given and the goal is to automatically (with no input parameter) segment the bundle into groups called regimes. A fully automated and unsupervised method based on statistical features called AutoPlait was designed by

Matsubara et al. [80]. AutoPlait is based on the concept of regimes, which the authors introduced into time series. A regime is a hidden state that describes the behaviour of a segment of a time series. A time series is generated by a Hidden Markov Model (HMM) of regimes that switches between/within them. To find the optimum set of regimes, the authors adopt the idea of compactness from coding theory. The goal is to generate the minimum number of regimes that best describe the HMM that creates the time series. To achieve this goal an iterative approach is proposed, which at each iteration breaks the currently found regimes into two regimes. The iteration process is governed by an objective function that describes each regime through coding theory (the number of bits used to code the time series, number of segments in each regime, length of segments, etc). In a closely related approach, Zhao and Itti proposed TSDecompose (TD), an information retrieval approach to find split points from a given time series [81]. TD is based on the definition of homogeneity for a time series. The homogeneity of a time series is defined through its symbolic representation of the time series [81]. The authors used Symbolic Aggregate Representation (SAX) [82] of a time series to transform a given time series into a sequence of symbols. The authors define the entropy of a time series by the total entropy of the symbols generating its SAX string. This string of symbols is then split into two substrings if the total entropy of the new pair of substrings is less than their parent. The dividing process stops when the decomposed tree is at its local minimum entropy. In Chapter 4, we use this method by filtering the generated substring using an upper and lower bound threshold over the length of generated segments to extract IoRs from time series.

The main drawback in these methods is that they are highly dependent on the source of the data stream, and thus can be hard to generalise [70]. Another drawback of statistical pattern mining techniques is that these techniques need to observe a substantial amount of data before they can detect changes in the underlying statistical pattern. This dependency on observing a substantial amount of data makes them unsuitable for real-time

problems where making a fast decision is crucial. It also stops these methods from detecting short segments where the distribution changes for a short period of time compared to the length of the time series.

Subspace tracking Subspace tracking is an approach that can be used for online time series segmentation, which falls into the class of derived metric boundaries. In this approach, segments can be well defined when there is little randomness in the interval as compared to the rest of the data. These algorithms estimate the stochasticity in an interval and look for segments with significantly less randomness. Fancourt and Principe [83] and Lin et al. [84] used subspace tracking in a specific domain for human motion recognition. However, these methods are highly susceptible to the presence of noise, i.e., a small perturbation inside the IoR can prevent these algorithms from differentiating the interval from the rest of the data [85].

Segmenting by motifs Motif discovery algorithms are another set of methods for time series segmentation, which search for physical boundaries [86, 87, 88]. A motif is a unique recurring pattern in the time series data. Recurrence was first formally defined by Henri Poincaré in 1890 [89]. Later in 1897 Eckmann et al. introduced the Recurrence Plot (RP), which is a binary image that can visualise recurrence in a dynamical system [90]. Structure in the plot may result from periodicity of the system, and can thus be used to detect recurrence in systems [91].

The most recent motif discovery method is Matrix Profile (MP), which can detect motifs from time series in an unsupervised manner, proposed by Yeh et al. [92]. MP is a vector such that at each point in the time series, it stores the Euclidean distance between the subsequence of length m starting from that point, to its nearest neighbour. MP searches for subsequences of fixed length m, and aims to find the closest subsequence to each subsequence of length m [92]. MP drops to zero during an interval of recurrence. Thus, to find intervals of recurrence in a time series we can search for points on the

MP with values smaller than a threshold for an interval. The main problem is that MP depends on its input parameter m to find the IoR. Gharghabi et al. used MP to design an arc counting method that considers each repeat as a motif, and tries to find repetitive regions from a time series [93]. Mirmomeni et al. showed that both MP and the arc counting method are highly sensitive to their key input parameter m for finding consecutive repeating regions in a time series [94]. They demonstrated that for each different region we must use a different m, which prevents this method from being a one pass and real time algorithm. They designed a method for finding the best m to identify a repeating region from a time series.

2.2.4 Evaluating trainees' performance from a time series—A time series analogy problem

To provide feedback on exercises using wearable sensors, we need to find mechanisms to evaluate the physical movement from the time series data. As discussed earlier, exercises are performed differently by trainees mainly to achieve various personal goals. Thus, there is no one correct way of performing an exercise that can be captured from the signal features of the time series data. As a result, research has been focused on finding methods to compare the observed time series data to some ground truth that has been captured in advance. In this section, we review computational methods for comparing time series data.

Time series analogy is the process of comparing pairs of time series data with each other. The comparison returns a value that describes how similar/dissimilar the time series are. These techniques can be divided into two main categories:

1. Analysing the time series by extracting features from the series, and
2. Analysing the time series by their shapes.

In feature extraction, the goal is to transform the time series into a feature space

based on statistical features such as the mean and standard deviation. These methods are often based on the feature extraction representation of time series discussed in Section 2.2.2. The time series is then processed as a vector in the reduced feature space. Modelling shape of a time series statistical models (ARMA) [95, 96, 97], extracting the most representative coefficients of a cosine transformation [98] and grouping patterns based on principal components using singular value decomposition (SVD) [99], are methods that reduce the time series dimension using feature extraction methods. These features are passed to learning models for further analysis. Note that in some learning models, feature extraction happens implicitly. For example, passing a spectrogram into a convolutional neural network results in a network to learn complex features from the frequency domain of the time series data. Although these methods improve processing time for large data sets by reducing the dimension of the time series into just a few features, they often lose the shape structure of the time series during the feature extraction process, a feature that is important to capture when analysing weight training performance from IMU sensors. In the problem statement, we mentioned that each performance of an exercise within a set results in a signal. The changes in the shape of these signals is the unit of comparison for our problem, i.e., to assess weight training performance from a time series stream.

Shape analysis of time series considers the overall waveform to be more interesting than each individual value in the series [100, 101, 102]. As a consequence, shape-based algorithms that quantify the similarity between pairs of time series search for common patterns in the shapes formed by peaks and troughs. This category of time series analysis can be divided into two subgroups: (1) Discrete time series, and (2) Continuous time series.

Discrete time series analysis refers to a family of methods where the time series is described through a few limited discrete values (symbols), which we described in Section 2.2.2 (Fragmented representaion of time series). Piecewise aggregation of the time series [57, 103, 56] and discretization of the curve to symbolic representations

through thresholding (SAX) [58, 104] are examples of methods that segment a time series. Computing a consensus is often used as a method to extract the average shape from a set of time series represented in a fragmented form [105]. A consensus is calculated by comparing a set of time series point-to-point, and accepting the most frequent value at each point from the aligned time series. Discrete time series analysis is limited by the number of symbols available to represent the time series, which limits the accuracy of the representation of the waveform.

Methods that work directly on time series values usually capture the most complete shape details of a waveform. The time series in these problems is represented by continuous measurements. The simplest form of continuous analysis is to align a pair of time series in a point-wise manner and calculate the *Euclidean distance* (ED) between them [106]. In this approach we can assume each time series is a vector in space as described by Buza et al. [107]. Then, the distance between two points is the vector distance between the points in the mapped space. Using the first few Fourier coefficients [106], reducing the time series to the first few Haar wavelet transformation coefficients [108], or modelling the curve using Chebyshev polynomials [109] are some early models that tried to map pairs of time series according to simpler models, such as the underlying frequencies of the two time series. More advanced techniques are based on the elasticity of time series or correlation of a pair of time series. These two techniques have been widely used in finding time series analogies which we describe next.

Elastic based time series analogy

In this technique, the goal is to find the best map between the peaks and troughs of two time series, i.e., different parts of time series with different lengths can be aligned with each other. Let $U = \{t_1, t_2, ..., t_m\}$ and $V = \{t'_1, t'_2, ..., t'_n\}$ be a pair of time series with length m and n respectively. Let $Dist$ be the distance function that computes the distance between U and V. The algorithm that tries to find the minimum value for $Dist$

by mapping parts of these pairs to each other is a dynamic programming approach called Dynamic Time Warping (DTW). DTW was first designed for speech recognition in 1971 by Sako and Chiba [110] and since then has became a popular method for time series matching. In DTW, $Dist$ is calculated by $D_{m,n}$ such that $Dist(U,V) = D_{m,n}$ where $D_{i,j} = d_{i,j} + \min\{D_{i-1,j}, D_{i-1,j-1}, D_{i,j-1}\}$ and $d_{i,j}$ is a distance function between t_i and t'_j. Since then there have been numerous variations to DTW. For example, computing $Dist(U,V)$ is computationally expensive. Pruning the search space for finding the best warping [111] and approximate indexing [112] are some of the early methods proposed to compute DTW efficiently. A parallel algorithm for calculating DTW was introduced by [113].

Keogh and Pazinni showed that DTW can result in misalignment, especially when a very small region such as a single point from one time series is mapped to a big portion of another time series [114]. They proposed a probabilistic approach that considers regional shapes as high level features for mapping a pair of time series. Ratanamahatana and Keogh proposed the idea of constraint time warping that set a bound on the search space [115]. In 2017, Cuturi and Blondel introduced softDTW, which used smoothed time series to propose a differentiable loss function for DTW [116].

The longest common subsequence (LCSS) distance was proposed to find the best common subsequence of two time series [117]. LCSS differs from DTW in that it does not need to map all the subsequences of two time series, and thus has more freedom in comparing two time series. Adaptations of encoding methods from compression theory [118], and frequency of occurrence of similar patterns in two time series [119] are other elasticity based approach that have been proposed for time series distance measures.

Correlation based approaches

Correlation based approaches are designed on basis that the time series is generated by sampling a continuous waveform, thus it is a signal. Consequently, correlation for time

series is defined in the same way as it is defined for waveforms, i.e., for two real valued time series U and V cross-correlation is denoted by $\circledast$ and defined by first extending the time series to infinity by zero padding them and then calculated as $U(t) \circledast V(t) = \int_{-\infty}^{+\infty} U(t+\tau)V(t)dt$. Use of correlation in time series similarity goes back to 1948 when Quenouille studied how a test of significance can be applied to time series data through their cross-correlation [120].

The cross-correlation of a pair of waveforms has a pseudo-metric property. This property of cross-correlation became popular for analysing time series when Podobnik and Stanley showed its effectiveness for general time series data [121]. Wachman et al. used this pseudo-metric property to find the similarity between time series generated from star brightness observations [122]. Paparrizos and Gravano used the pseudo-metric property to design an efficient K-Means clustering algorithm for time series data [61].

2.2.5 Computing the correct performance of an exercise—A time series averaging problem

Time series averaging is the problem of computing a time series (centroid) that represents all of the time series in a given set. The centroid plays a key role in both applying tests of significance to time series data sets, as well as prototyping a set of time series. In tests of significance, anomalies can be detected through their distance from the centroid. In prototyping, a centroid is defined as the representative of a set that can be used in further time series analysis to achieve greater efficiency. For example, to determine whether a given time series is from a particular group, a common approach is to compare the new time series to the centroid of the set and assign the time series to the group with the closest centroid.

DTW Barycenter Averaging (DBA) is an averaging method based on Dynamic Time Warping (DTW). Gupta et al. showed that finding the global optimum prototype using DTW is an NP-Complete problem [123, 124]. This occurs because finding the optimum

prototype is sensitive to the order of presentation of the inputs. Petitjean et al. introduced DBA, an iterative heuristic approach to overcome the ordering problem [124]. The initialisation of DBA chooses a random curve in a group. The objective function that monitors the progress of DBA and K-Means clustering is the sum of squared errors between the prototypes and the input data. Let $X = \{x_1, \ldots x_N\}$ be a set of unlabelled waveforms in feature space R^p (i.e., each x_j is a time series of p measurements); and let $V = \{v_1, \ldots v_k\}$ in R^p be a set of k prototypes for the waveforms in X. Each V_j represents a subset X_j from X. Then the within-group sum of squared errors (WGSS) between the input data and the prototype, with respect to any vector norm $\|*\|$ on R^p is

$$WGSS(V) = \sum_j \sum_i \|x_{ji} - v_j\|^2 \tag{2.1}$$

where x_{ji} is a member of X_i. In each iteration, DBA computes a prototype that lowers the within-group sum of squared errors. In their most recent approach, Petitjean et al. introduced the idea of finding multiple prototypes for a given group. In this approach, the input space is divided into multiple sub-regions and then a prototype for each sub-region is computed [125]. This technique performs better for classification/clustering of data than the classical K-Means method. The sub-partitioning tries to produce convex hulls for which the average is well defined (i.e., the average always appears inside the convex hull of its generators).

Morel et al. introduced constrained dynamic time warping for averaging time series (cDBA) [126]. Their method is very close to DBA with an extra constraint on mapping through DTW. The authors showed that in cases where zero-mean normalisation cannot be performed over the data, DBA fails to create an accurate prototype. They proposed the use of locally constrained DBA (cDBA) defined by Muller [127] to improve the DBA prototype in non-normalised data sets. They empirically showed that the prototype computed using locally constrained DTW improves the DBA results in a classification

task.

Papparizos and Gravano introduced Shape Extraction (SE) for computing a prototype from a set of time series [61]. Their scheme is based on the correlation distance of time series data. They attempt to find an optimal prototype for a cluster using signal correlation maximisation. The main idea is based on a convolution operator that creates an inner product space and thus can be used as a pseudo-distance. In this method, each z-normalised member of a given class is considered to be a vector in R^n where n is the number of features in the time series. In this approach, signal correlation is used as the distance between pairs of vectors. Let $X = \{x_1, \ldots x_N\}$ be a set of unlabelled waveforms in feature space R^n that generates a matrix of size $N * n$. Thus, for a given subset of vectors in R^n the eigenvector associated with the biggest eigenvalue defines the dominant orientation of the spread of the vectors in R^n. Since every vector is z-normalised, the z-normalised eigenvector associated with the biggest eigenvalue of the given subset can be used as the prototype of the subset. Papparizos et al. showed how to compute this eigenvector using a few linear transformations [61].

2.2.6 Overview of current time-series analysis methods for addressing real-time and online exercise monitoring

Table 2.1 summarises the available methods for tracking exercises from time series data in terms of the extent to which they satisfy our design goals. The table shows that the state-of-the-art techniques all have limitations in terms of efficiently and effectively detecting and tracking exercises using data from a given wearable sensor to provide feedback on weight training activities.

Table 2.1: Comparison of current state-of-the-art methods for identifying and tracking weight training activities from incoming streams of data from wearable devices in terms of design goals from Chapter 1

	Real time Segmentation	Online Segmentation	Resource Efficient	Unsupervised	Adapt to Variations	Handles Environmental Noise
Interval [71, 69, 59]	Yes	Yes	Yes	Yes	No	No
CPD [72, 73, 74, 78, 75, 76, 77, 80]	No	Yes	Yes	Yes	No	No
Subspace [83, 84, 85]	No	Yes	No	Yes	No	No
Motifs [86, 87, 88, 92, 93, 94]	No	No	No	Yes	Yes	Yes

2.3 Timing the feedback: When to provide the feedback to the trainee?

The timing of feedback plays a key role in deciding what information we need to provide to the trainee. Personal trainers who monitor trainees offline want detailed information of their trainees' performance. For example, routine intensity and performance accuracy as well as recovery period measurements are essential information that personal trainers access to design their trainees' routine. In offline scenarios, more computationally intensive analysis can be used to provide detailed analysis of the trainees' performance to their personal trainers. However, trainees also want to know if they are performing exercises safely, i.e., they want to know if they are maintaining correct posture while performing their exercises. In this regard, trainees need to receive their feedback as soon as possible while performing the exercises to avoid injuries. Machine learning techniques to provide the feedback can be categorised into: (1) Offline methods, and (2) Online methods.

2.3.1 Offline feedback

Offline methods target the life cycle of personal trainers and their trainees. These methods aim to provide as much detail as possible to personal trainers about how their trainees performed their exercise routine over a period of time (often six to eight weeks). Current methods are based on manual logs that trainees record while performing their exercise routines, which can be discussed with their personal trainers when they meet. Pernik et al. designed one of the first machine learning techniques for wearables to define the intensity of each set performed by trainees [128]. They designed a hierarchical workflow that uses a SVM to first detect the exercise type performed by the trainee using accelerometer data generated from multiple wearable sensors worn by the trainee. The system then uses another SVM to identify the intensity of each detected exercise. They reported that their

system's exercise recognition performance is not consistent over all exercise types, i.e., the system detects some exercises with higher accuracy than others. Thus they suggest to use a fusion of classifiers to achieve higher accuracy, which makes their method more computationally resource intensive.

Challenges The main challenge that these methods face is to collect the ground truth for evaluating trainees' performance during the exercise routine. They often use questionnaires or user feedback to define the evaluation ground truth. However, this type of evaluation is shown to be highly correlated to users and does not generalise well [129]. Thus, more systematic effort is required to collect the necessary data to address this type of evaluation thoroughly. In this regard, Capecci et al. collected the first data set that contains physiological data of the activities of both healthy subjects and patients, which was mapped with a clinical score that was assigned to each activity by clinicians [130]. This approach provides an important first step towards generating valuable insights on trainees' performance in an offline scenario.

Strength and weakness The main strength of offline methods comes from their ability to provide detailed analysis of trainees' performance automatically. The information that these systems can provide is independent of a user's feeling or mood on a particular day, which provides more objective overall evaluation of trainees. Collecting this information autonomously is the second benefit that these systems provide to their users. Users are not obliged to collect their data manually and can focus their full attention on their routine.

The main weakness of these systems is that they cannot warn their users during their routine. The source of this shortcoming is the type of machine learning technique that these systems use to analyse the data, which cannot be used in an online and real-time environment.

2.3.2 Online feedback

Online feedback is any feedback that is provided to the trainees during their training. This type of feedback is used for two main purposes:

(1) Teaching purposes, and

(2) Warning purposes.

Feedback for teaching purposes is designed to teach trainees how to perform an exercise. These systems often consider trainees as beginners who want to learn a specific routine. Visual feedback for teaching purposes has gained considerable attention. MotioMA was one of the earliest systems to address this problem [18]. In MotionMA, the system learns an activity through an expert demonstration in front of a camera. After learning the exercise, users can monitor their performance in front of a mirror where the system reflects the user's performance by mapping it into the expert's performance that it learnt in the first step. The system can detect incorrect movements from the mapping and suggest improvements.

Although visual feedback to beginners can be highly effective and improve their learning speed, it does not address the needs of more advanced trainees. These trainees are people who aim to master the performance of exercises. However, they may fail to maintain their posture or performance due to the pressure of their training routine (these types of pressure were discussed in detail in Section 1.1.1). In this scenario, the solution is to monitor the trainees in real-time and evaluate each exercise's performance with respect to the trainee's best performance. In this way, a system can monitor the trainee without imposing any unwanted limitations, such as different postures for a particular performance, by first trusting the trainee, and second learning the movement pattern from the trainee.

Strength and weakness The main strength of online feedback is that these systems can play an essential role in avoiding injuries by trainees. The need for both warning and teaching systems arises from the fact that incorrect performance of an exercise and incorrect posture can result in injuries that may have life-long impact on the trainee. Thus, providing online and real-time feedback to trainees can help them to undertake their exercise safely and reach their goals more easily.

The main weakness of online feedback is a lack of in-depth analysis of each performance. This shortcoming arises from the need to provide feedback as soon as possible, which makes it infeasible to perform a thorough analysis of the data available to provide a feedback. As always, there is a trade off between the feedback timing and the amount of data that is possible to analyse to provide detailed feedback.

Challenges: The main challenge for online and real-time feedback comes from the amount of data to be analysed and the computational resources available to generate the feedback. For example, if a system is to provide a warning signal to its trainees while performing an exercise through wearable devices, it faces the challenge of designing resource efficient machine learning techniques that can run on a wearable device. In particular, wearable devices have limited computing resources and energy storage, and need to be durable in harsh environment.

2.4 Summary of state-of-the-art technologies for our design goals

Table 2.2 shows the current state-of-the-art technologies with respect to our design goals. The table shows that wearable technologies (in the form of gloves or smartwatches) are the only technique that are capable of being ubiquitous, which illustrates the potential that wearable technologies have in meeting the design goals. In the next chapter, we

begin our research into machine learning techniques for providing feedback on wearable devices for weight training, by describing our overall workflow for unsupervised learning in weight training.

In the next chapter, we begin our research into machine learning techniques for providing feedback on wearable devices for weight training, by describing our overall workow for unsupervised learning in weight training activities. We design the first workflow that detects the deviations in the performance of weight training exercises from a standard form without the use of labelled data. Our aim is to design machine learning steps that need to be taken to achieve the first workflow to detect weight training movements without the use of labelled data, i.e., an unsupervised method. We shed light on why current machine learning techniques are incapable of fulfilling our design goals. We then focus on designing the learning techniques for each of the building blocks of our workflow with a focus in addressing our design goals discussed in Table 1.1.

Table 2.2: Comparison of current state-of-the-art technologies for identifying and tracking weight training activities in terms of design goals from Chapter 1.

	Device	Real time Detection	Real Time Qualification	On a wearable	Unsupervised	Adapt to variations	Environmental Noise	Ubiquitous
GymCam [31]	Camera	Yes	No	No	No	Yes	Fail if obstructed	No
MotionMA [18]	Kinect Camera	Yes	Yes	No	No	No	No	No
YouMove [33]	Augmented Reality	No	Yes	No	No	No	No	No
SmartMat [35]	GYm Mat	No	No	No	No	No	No	Semi
SmartSurface [36]	Surface Tiles	No	No	No	No	No	Yes	No
SmartGlove [39]	Gloves	Yes	No	Yes	No	No	No	Yes
RecoFit [6]	Smartwatch	Yes	No	Yes	No	No	Only seen noise	Yes
NuActive [41]	Smartwatch	No	No	No	Semi-supervised	Yes	No	Yes
Maxxyt [7]	Smartwatch	Yes	No	Yes	Yes	Yes	No	Yes
MilFit [8]	Smartwatch	Yes	No	No	Yes	No	Yes	Yes

Chapter 3

A workflow to detect incorrect performances in weight training activities

3.1 Introduction

In this chapter we propose a general workflow to detect incorrect performances of a weight training exercise. This workflow provides an overall framework and context in which we can investigate the underlying machine learning challenges that are addressed in the subsequent chapters. Here, we use prerecorded time series data from weight training exercises, and we assume that the analysis is not limited by computing power. Thus, the focus in this chapter is to show how to detect incorrect performances from time series data recorded using an IMU sensor worn by a trainee.

As discussed in Section 2.2.2, designing a system based on IMU technologies brings other technical challenges that arise from the nature of these sensors. The main challenge is to infer the whole movement from the motion measurements. In the IMU we are dealing with a set of motion sensors that monitor each action independently. Thus, the

body movement needs to be derived from a set of motion sensors; in other words, we are inferring the body movement from the measurements of multiple motion sensors.

The second challenge is to identify each exercise within a set individually. This is essentially a problem of accurately segmenting the underlying time series data into exercise segments. We revisit the dataset recorded by Velloso et al. to demonstrate how an accurate segmentation can improve recognition performance [18].

In summary, we propose a workflow for performance error detection in weight training exercises (see Figure 3-1). Thus in this chapter we focus on the following issues:

1. A mathematical model for repetition segmentation based on a generalisable mathematical feature from weight training exercises, along with its concrete implementation and evaluation;
2. A method that learns a prototype of an exercise from IMU sensor data;
3. A statistical method for identifying incorrect executions based on deviations from the exercise prototype;
4. An evaluation of our approach on a publicly available dataset that shows that our system is able to identify incorrect performance with 98% accuracy.

3.2 Workflow

Figure 3-1 gives an overview of the workflow for detecting incorrect performance of an exercise solely based on observing correct performances of the exercise. The workflow has three main components. First, the workflow starts by segmenting the received time series to extract each exercise repeat from a set. Second, it extracts a prototype that describes the correct performance of an exercise and derives the distribution of correct performances. Third, it evaluates each new performance of the same exercise with respect to the computed distribution and evaluates its correctness. Next we explain each

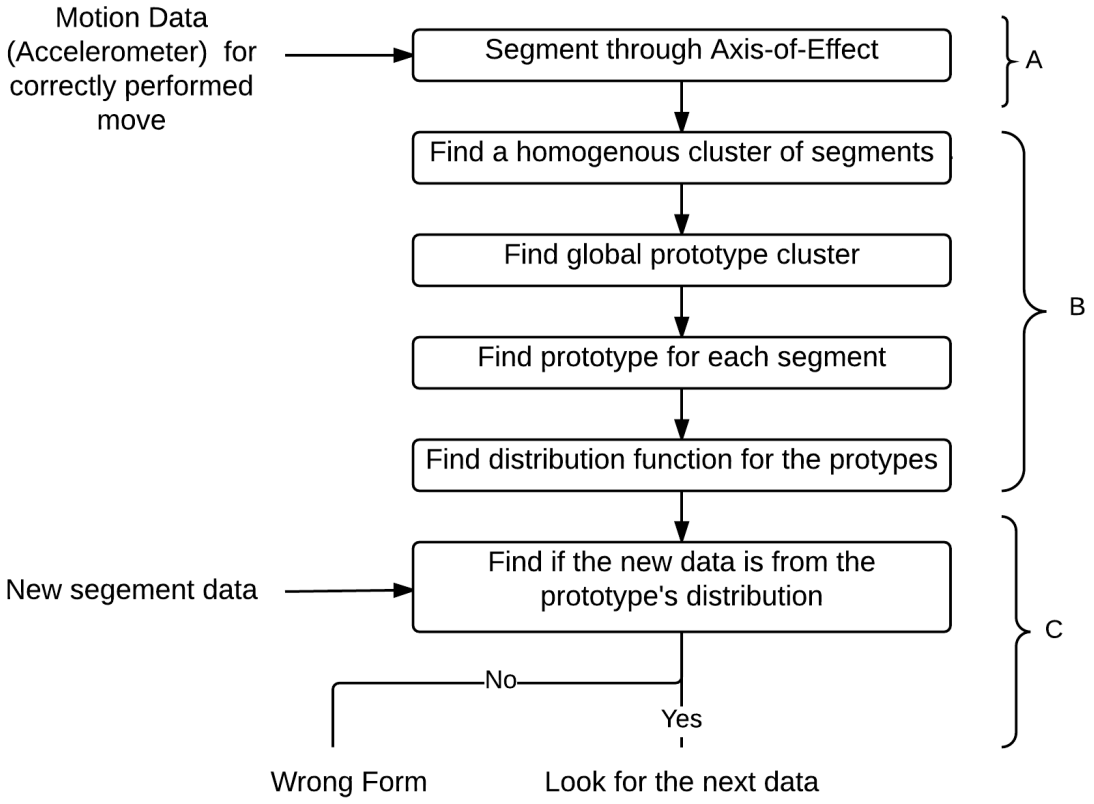


Figure 3-1: Workflow for finding incorrect moves in a weight training exercise.

component in more detail.

3.2.1 Segmenting repetitions

In this section, we revisit the dataset recorded by Velloso et al. to demonstrate how an accurate segmentation can improve the mistake recognition performance [18]. Segmentation here refers to the process of finding each individual repetition of an exercise in a given time series. Since our focus is on each individual repetition we need to find a way to correctly extract each repetition from a given accelerometer time series data stream.

Velloso et al. studied the possibility of classifying form correctness in weight training through unilateral biceps curls [18]. They took a supervised learning approach, by recording both the correct execution of the exercise, as well as common types of mistakes. Because we use their dataset to evaluate this approach, in this section, we briefly describe

(a) Starting position

(b) Biceps movement

Figure 3-2: Illustration of unilateral biceps curl performance.

the dataset and propose a new segmentation approach that improves the recognition accuracy using the same classifier used by those authors.

Dataset The dataset of Velloso et al. includes the data from six participants performing 10 repetitions of a unilateral biceps curl exercise with five variations [18]. The dataset is publicly available in the UCI Machine Learning Repository[1]. A unilateral dumbbell curl is a weight training exercise focused on strengthening the biceps muscle. The description of the exercise is as follows and is illustrated in Figure 3-2:

1. Stand with a dumbbell in each hand.
2. Keep the upper arm stationary, while bringing one of the forearms up until it reaches its maximum bend.
3. Return the arm to its original position slowly.
4. Repeat the same move with the other arm.

The authors defined four common errors that can occure during this exercise as follows: Class B: Moving upper arm to the front. Class C: Lifting the dumbbell half way up and return. Class D: Lowering the dumbbell halfway down. Class E: Pulling the forearm

[1]Available at: `https://archive.ics.uci.edu/ml/datasets/Weight+Lifting+Exercises+monitored+with+Inertial+Measurement+Units`

with the help of the hips at the start to lift the dumbbell. The correct form of the exercise is labelled Class A, accordingly. For each class label, they asked six participants to perform the exercise according to its class label specification. They collected the data using four sets of sensors placed on the glove, upper arm, dumbbell and belt. Each sensor set contains one three-axis accelerometer, one three-axis gyroscope and one compass. Each participant performs 10 repeats of the exercise for each label. For the detailed explanation of the dataset collection method see [18].

Pre-processing Any motion sensor shows some degree of white noise, which comes from the nature of the sensor. White noise appears as small perturbations around the actual value. Any successful analysis of the data must first remove this noise from the data [131]. In the literature the smoothing effect of Kalman Filters has been widely used as a low pass filter that can remove white noise with high accuracy [132, 133]. In this project, we used a density based Kalman Filter implementation [134]. Since the focus is to remove white noise, we configured the filter with low sensitivity to the current read and high sensitivity to previous reads by setting the deviation of the current reads to be 10 times the deviation of the previous reads. The result is shown in Figure 3-3.

We draw the readers' attention to the peaks in the raw data. The accelerometer shows a very sudden, high acceleration and drop, which is due to the effect of stopping the dumbbell. From Figure 3-3 we can see how a Kalman Filter reduces this effect and smooths the result.

Since the data is published separately, for consistency we checked all the labelling to verify that the labels are correct. We found issues with the correct execution of two of the participants. By definition, the label-A data should have a steady pattern from the belt accelerometer data. However, in two cases, the participants incorrectly moved their hips during the exercise, as illustrated in Figure 3-4. We therefore discarded the data from these two participants from the subsequent analyses.

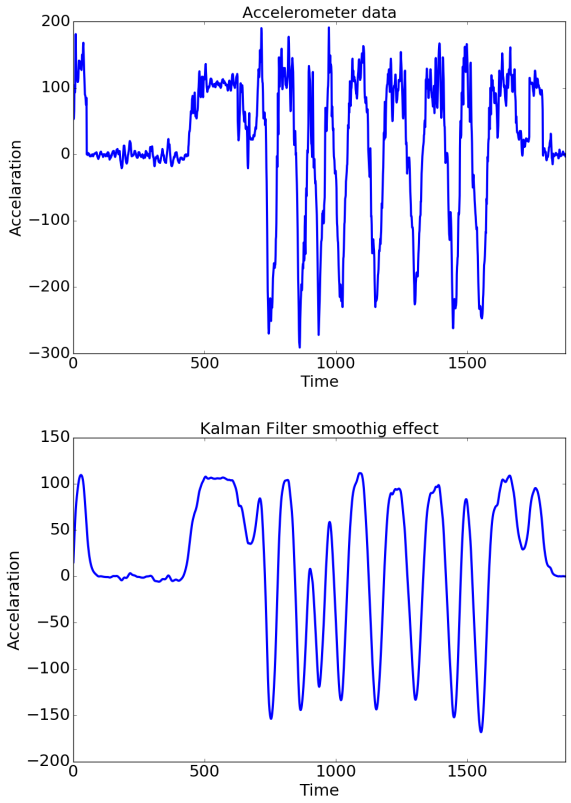


Figure 3-3: Smoothing accelerometer data with a Kalman Filter. Top figure shows the raw data from an accelerometer. Bottom figure shows the result of applying Kalman Filter on the same data.

Segmentation In this section we show how to utilise knowledge of the general domain of weight training to indicate the exercise repetitions. In Section 1.1 we discussed that weight training exercises are repetitive activities where trainees perform the same move for a few repetitions, i.e., the move starts from a starting point, follows a path in space and returns to the starting point again. Each exercise set contains three-to-many of the same repetitions. Therefore, the underlying time series received from motion sensors sampling the unilateral biceps curl show a cyclical pattern in the time series. An example for acceleration data from a unilateral biceps curl can be seen in Figure 3-5.

As a result, correctly identifying the move using the motion-graph requires finding its starting point in the acceleration graph. To formulate a move, let f be the time-movement function showing the path for the move in space. Thus f is a function of the

change-of-position in the x, y and z directions, i.e., $f = g(x, y, z)$

As a result, function g is defined to describe the move. Since finding the actual move is highly dependent on all three dimensions and very sensitive to any noise/error in the data, we followed the method described in [135]. They showed that in weight training exercises, motion can be captured mainly from a single axis in space. We call this axis the *axis-of-effect* (AoE). For example, the y-axis is the AoE in the unilateral biceps curl exercise (Figure 3-6).

The unilateral biceps curl in the AoE direction starts with a positive acceleration to lift the weight up. Then, it follows a negative acceleration, which results in stopping the dumbbell at the peak. It is then followed by a negative acceleration pattern to bring down the dumbbell, which follows a positive acceleration to bring the dumbbell back to the steady point. Therefore, function f can be estimated by $g(t_{AoEi})$ where t_{AoEi} is the i^{th} sampled read value from the underlying time series received from the accelerometer in the AoE direction, i.e., $f \approx g(t_{AoEi})$. Considering the unilateral biceps move, we are searching for points in the function f where the function has reached its

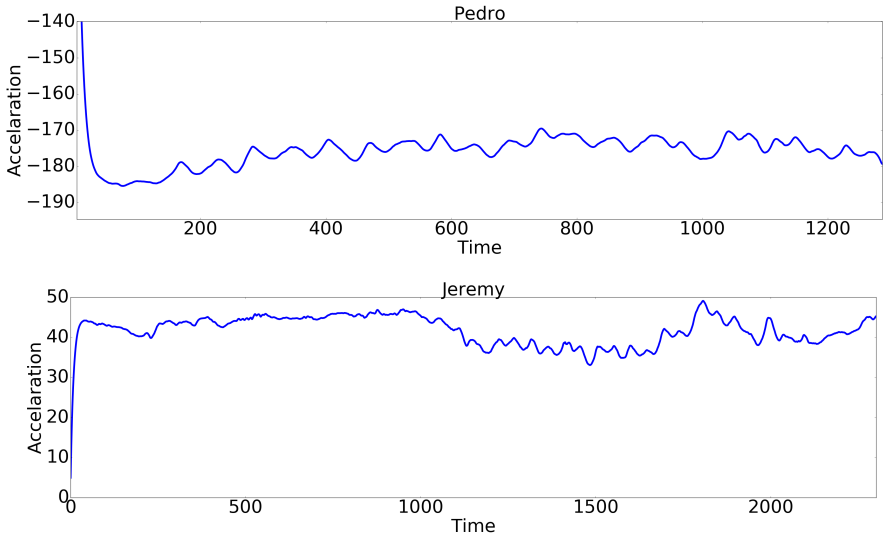


Figure 3-4: High perturbation area for the subject Jeremy[2] makes his class label "A" invalid. For a correct label for class "A" we expect a steady series like the one in the top figure. Note that the actual repeats start at around time = 800 for Jeremy.

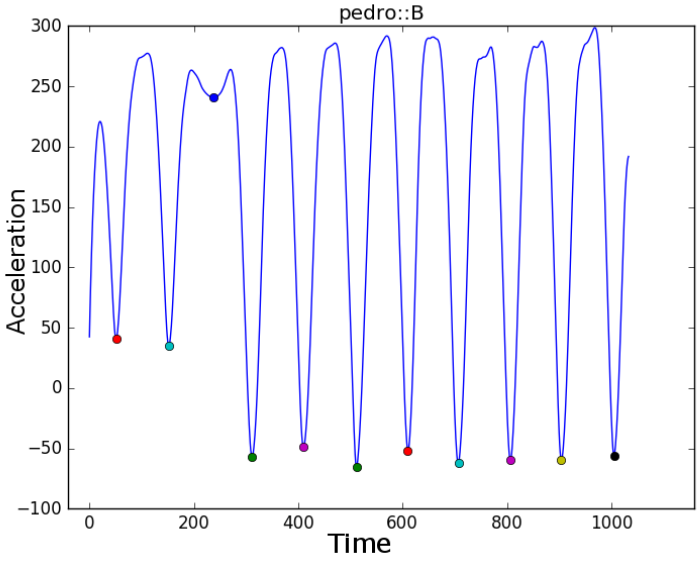


Figure 3-5: Motion sensors show a repetitive pattern for weight training exercises. In this figure, the repetitive pattern is clear from the accelerometer attached to the trainee's forearm while performing a unilateral biceps curl.

Figure 3-6: y-axis is the Axis-of-Effect (AoE) in unilateral biceps curl. The main motion occurs in the y direction (drawn in red).

maximum/minimum values. That is:

$$\frac{df}{dt} = 0 \tag{3.1}$$

Using the chain rule:

$$\frac{df}{dt} = \frac{dg}{dt}\frac{dt}{di} = 0 \tag{3.2}$$

Since changes in acceleration can describe the changes in movement, we can estimate the optimum points by only considering $\frac{dt}{di} = 0$. That is:

$$\frac{df}{dt} \approx \frac{dt}{di} = 0 \tag{3.3}$$

We look for points in the acceleration time series where the derivatives went to zero, or equivalently we search for points where the derivatives of the acceleration time series changes sign. Since we only need to find the start points of each move, we only look for minimum points on the acceleration-time graph, which lets us segment the repetitions (indicated by dots in Figure 3-5).

To evaluate the accuracy of our model, we manually annotated the acceleration time series and compared the manually annotated segments with the automatically detected segments. Table 3.1 shows the performance of our model for finding the repeats in the acceleration time series.

Table 3.1: Average precision and recall for segmenting the biceps curl routine using change point detection

	Precision	**Recall**
Average	0.965	0.82

The result shows that our algorithm finds the correct segmentation points with high precision. The recall value shows that the algorithm finds more segments than necessary. Filtering the points by the average repetition size improves the recall to 90%.

Classification In Velloso et al. [136], the authors reported the best classification accuracy achieved by segmenting the data using a fixed sliding window of length 2.5 seconds.

This result comes from the fact that each repeat takes around 2 seconds. Therefore, setting the window size to 2.5 seconds captures the entire move. However, a fixed window size can result in two main problems. First, a fixed window size captures overlapping repetitions, resulting in missing the start and end part of a repeat. Missing the start and/or end of a repeat stops any system from correctly alerting users as soon as they start deviating from a correct form. Second, a fixed window size is highly dependent on the person and the exercise. For example, if the user performs the exercise too quickly, a fixed window might capture two or more repetitions. Given that our unit of analysis is each individual repetition, it is crucial to capture the whole repetition with no extra data.

Table 3.2: The average precision of the classification task as a percentage. The top row is from data segmented by our method. The bottom row is from a fixed window size.

Algorithm	A	B	C	D	E
Optimum point	57.3	56.3	56.7	56.4	56.8
Fixed window	52.1	54	53.5	53.7	53.2

For comparison, we applied the same classifier proposed by Velloso et al., only using our segmentation method. The result is presented in Table 3.2. The result shows that our proposed algorithm not only provides a method that can easily be generalised, but also boosts the classification accuracy.

Discussion These results show that in a supervised learning scenario, where we have data for the correct execution of an exercise, as well as data for each possible repetition, segmenting the data using windows that precisely match the repetition improves the classification performance. However, it is unlikely that in a realistic use case we would have data for every possible mistake. Given that an incorrect form might result in lifelong injury, the stakes for the problem are high enough that we need a system that can robustly detect previously unseen mistakes based only on the correct form of the exercise. In such systems, a model should be created from the correct moves. After learning the correct move, every move is compared with the ground truth model to detect deviations from the

model. Designing such a model is not a straightforward task. There are many parameters involved in a ground truth model such as the height of the person, the weight used for the exercise, the duration of the repeat, etc. In the next section, we show how to derive a personalised model from the correct moves in the exercise, as a ground truth model for the unilateral biceps curl.

3.2.2 Deriving a ground truth model

The two scenarios we presented above suggest that at some point the trainee will be able to demonstrate correct performance, either because she/he is under the supervision of a trainer or is using a lighter weight. Using this assumption, we can derive a user-dependent ground truth model specific to the trainee's needs that they can use in subsequent repetitions (when the trainer is absent or with a heavier weight) to evaluate their performance.

Data collection In this phase the personal trainer makes sure the person is capable of correctly performing the exercise and initiates data collection. The data labelled 'A', in the dataset of Velloso et al. is used as the data associated to the correct performance of the exercise.

Pre-processing As discussed earlier, the recorded data for an exercise routine includes multiple repeats of the same exercise. However, each routine starts and ends with recordings, which are usually related to releasing or carrying the weight, not related to the actual routine. To clear Class A segmentation, we used a clustering algorithm to put the segments with high similarity into the same group. This task is very important because we can make sure the ground truth method is only generated from homogeneous segments and prune any anomalies from itself. We continued the clustering algorithm until we had a cluster of size "number of repeats - 2". This value is selected because we are aware that starting and ending segments might have extra movements that make

them different from the segments for the rest of the repeats. We used the single linkage clustering algorithm to cluster our segments [137]. For the distance measure in single linkage clustering, we used two measures DTW [115] and Shape Based Distance [61].

Ground Truth Prototype Deriving a prototype for time series data is a challenging task. The main issue is how we can map one time series to another. The main method for such a mapping is based on finding the minimum distance from mapping each point from one time series to another. Dynamic Time Warping (DTW) [127] has been traditionally used to perform this task. More recently, the K-shape [61] algorithm has been introduced with promising results. Both approaches derive a comparison based method to define a distance between set of time series data. By clustering the time series data, Paparrizos and Gravano showed how to design an algorithm that can find a trajectory which satisfies the minimum sum of distances between its points and all the other points in the associated time series cluster.

Paparrizos and Gravano showed that given a distance matrix, we can reduce the problem of finding the prototype for a cluster to a maximum likelihood problem where the eigenvectors of the Hessian matrix define the prototype for the cluster. They named their algorithm shape-extraction, which we adopt in this chapter. For a detailed argument see the original paper [61]. In this chapter, we use the same method for deriving a prototype trajectory from the correctly performed moves. We then use a statistical method to capture any deviation from this prototype.

To find the ground truth trajectory, we proposed Method 1. For each person in the dataset, we passed all the homogeneous segments clustered in pre-processing part from the class A dataset to the shape-extraction algorithm and find the ground truth trajectory (Personal GTT).

Finding anomalies Using Personal GTT, we first enumerate each segment from each person and calculate the associated trajectory for that segment. Then, for each point in

Algorithm 1: Finding Ground Truth.

```
1  Function ground_truth(C):
       Data: C: cluster of homogeneous segments for correct move performed by the person
       Result: Prototype for given cluster, the mean and standard deviation of the calculated
               prototype according to the member of the cluster.
2      globalTrajectory ← extract_shape(C);
3      trajectory ← [];
4      mean ← [];
5      sd ← [];
6      forall c ∈ C do
           shape ← extract_shape([c, globalTrajectory]);
           trajectory.append(shape);
7      end
8      forall p ∈ globalTrajectory do
           m ← mean(trajectory_p) // trajectory_p is the p^th value of all trajectories
           s ← standard_deviation(trajectory_p)
           mean.append(m)
           sd.append(s)
9      end
       return globalTrajectory, mean, sd
```

time in the trajectory set we find the mean and standard deviation among all calculated trajectories. Receiving any new segment for the person we find the trajectory for the new segment according to the related Personal GTT. We then compare each point in the new trajectory to be in the mean $\pm$ 3s.d. range. Three standard deviations is selected from the 3-sigma rule, which states that nearly all values from a distribution occur within the range of three standard deviations from the mean value [138]. If any point is found outside this margin from the new trajectory, we label it as a wrong form segment. See Algorithm 2.

Since anomalies may be found from different sensors — for example when the trainee is moving their hip, the belt accelerometer will detect the mistake — we applied the same algorithm (find-anomalies) for each time series from the available accelerometer sensors attached to the belt, arm and forearm. We define a deviation from the correct form as any deviation from any time series from any sensor. This way our system can also detect why the person is deviating from the correct form. For example, in the test study for the

Algorithm 2: Checking for anomalies

```
1  Function find_anomaly(N, G, M, S):
       Data: N: New segment;
       G: Global prototype from Algorithm 1;
       M: mean for G (from algorithm 1);
       S:standard deviation for G (from algorithm 1)
       Result: True if the new segment is an anomaly otherwise False
2      shape ← extract_shape([N, G]);
3      forall p ∈ shape do
           // shape_p is shape's value at index of p,
           // S_p is S's value at index of p,
           // M_p is M's value at index of p
4          if shape_p > M_p + 3 * S_p || shape_p < M_p − 3 * S_p then
               return True
5          end
6      end
       return False
```

unilateral biceps curl, the system tells whether the trainee is moving their hip or their arm when they were not supposed to. The system also detects when during the segment the user deviated from the correct form, i.e., whether it was in the first quarter of the move, in the middle or in the last quarter. These two sets of information not only help users maintain the correct form but also let personal trainers to design more personalised routines that consider the strength and weakness of each trainee. For a brief overview of the output of each step in the workflow see Figure 3-7. The workflow starts by collecting data for the correct performance of the move (sample data is shown in Figure 3-7a). We remove all the white noise from the collected data using a Kalman Filter and segment the data using our proposed segmentation algorithm (Figure 3-7b). We create a homogenous cluster from the segments created in part B. (Figure 3-7c). We derive the prototype for the ground truth performance using the homogenous cluster (red line in Figure 3-7d). We calculate the standard deviation for our prototype using the correct performance (dashed line in Figure 3-7e). Note that the homogeneity of segments in part C makes sure that only the correct performances of the exercise are considered for deriving the ground truth in part D, and not moves with an extra part such as the very first move in which the

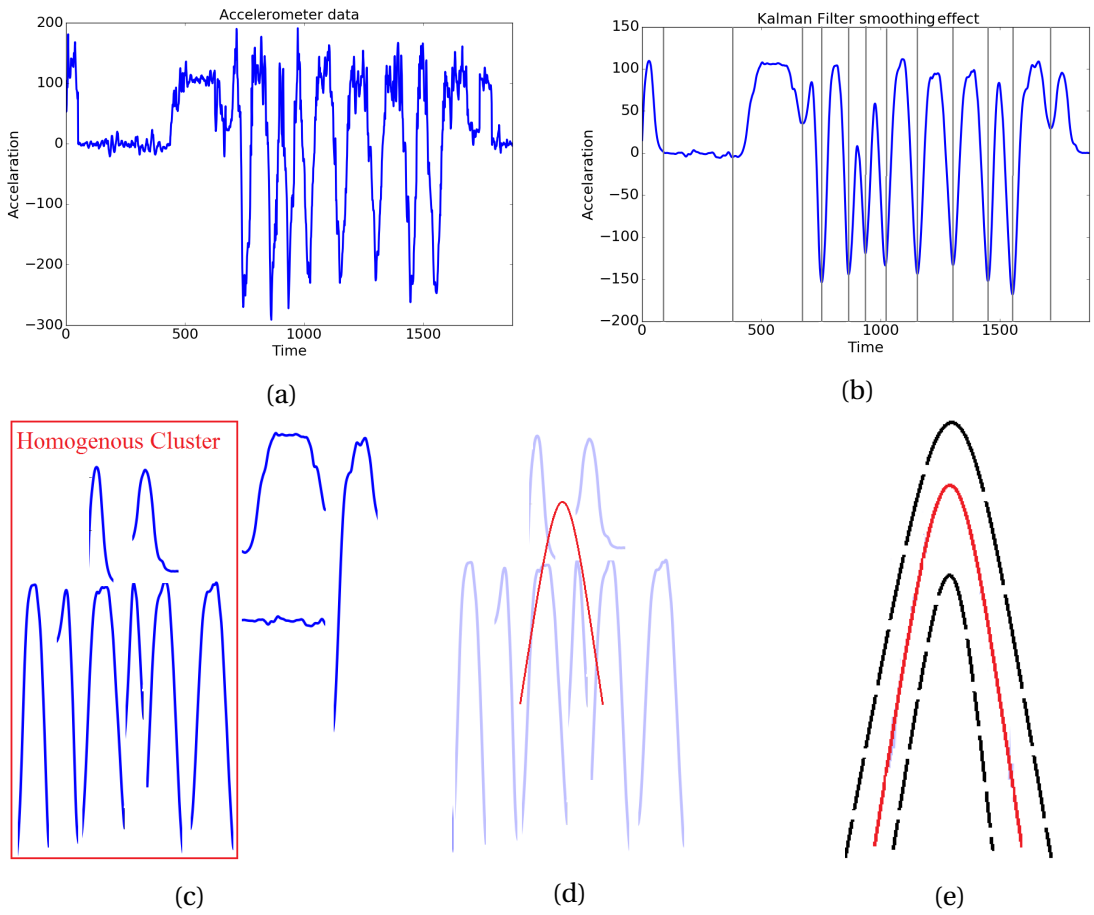


Figure 3-7: Outline of workflow for finding weight training anomalies. (A) Collecting raw data for correct performance. (b) Smoothing the raw data. (c) Creating a homogenous cluster of correct segments. (d) Deriving the ground truth (red line). (e) Calculating standard deviation for the ground truth (dashed line)

data contains the part where the trainee is picking up the weight at the very beginning of the routine.

Results To test our approach, we used the same dataset and two distance measures, namely dynamic time warping [115] and shape-based [61]. We considered class label A as the correctly performed class and used the rest of the classes as the test cases. For testing our ground truth trajectory, we manually labelled each correct segment in class A and feed them to Algorithm 2. The result can be seen in Table 3.3. Both algorithms can reliably detect mistakes for the unilateral biceps curl. However, it is clear that the shape based

Table 3.3: True Positive (TP) and False Positive (FP) rate for anomaly detection algorithm. The cells with bold font show the winner algorithm for detecting anomalies.

	Shape Base Distance		Dynamic Time Warping	
	TP	FP	TP	FP
Pedro	**41/42**	**0/10**	40/42	1/10
Carlitos	**44/44**	0/10	43/44	0/10
Charles	**44/44**	**2/10**	44/44	3/10
Eurico	40/40	**2/10**	40/40	3/10

distance is better able to find the trajectory. This is mainly because of the way the k-shape algorithm computes distances, which highlights the correlation among the points in two time series. Both algorithms have shown some false positives, which are mainly for the segments at the start or end of the routine. This is mainly because the start and end of a routine is very hard to correctly segment. It is often the case that the segmentation has considered an extra part at the beginning of the segment for the starting segment, or considered an extra part at the ending segment. These mis-segmentations result in false positives in our algorithm.

3.3 Discussion

In this chapter, we designed the first workflow that detects incorrect moves from learning only the correctly performed routine. We showed why correctly segmenting each repetition during a weight training exercise is important. The workflow starts by correctly segmenting the time series data using the data from an Axis-of-Effect accelerometer. The workflow then calculates a prototype from the exercise segments. Using the derived prototype, the workflow finds the distribution of the trajectory from mapping each segment to the prototype. Finally, for each new segment it checks whether the new data's trajectory to the ground truth is from the calculated distribution or not. If not, the algorithm rejects the segment and alerts the trainee of an anomaly.

3.4 Conclusion

In this chapter we demonstrate the ability to infer incorrect performance of a weight training exercise through the signals received from a wearable attached on the axis-of-effect. As discussed in Section 3.2.1, if a single sensor is placed on the axis-of-effect from an exercise as suggested by Mortazavi et al [135], any deviation from the correct form can be detected. Therefore, any exercise with a single axis-of-effect (AoF) can be analysed using our workflow. Examples of such exercises include, but are not limited to, squat (AoF upper leg), biceps curl (AoF wrist), shoulder press (AoF wrist) and deadlift (AoF upper leg). Note that the workflow is accurate for any exercise as long as the sensor is located on the correct part of the body (the axis-of-effect). We have designed the necessary steps to be taken to find these anomalies in an offline environment.

A successful analytical approach for providing insightful feedback to weightlifters must solve the following challenges:

1. Identifying the interval of interest: Capturing time intervals where the trainee is performing an exercise is a critical step in analysing the routine. These intervals are usually surrounded by periods of rest where the trainee is free to walk around or stand still.
2. Quantifying the quality of the performance: By detecting the interval of interest, a successful analysis must be able to give feedback about how the trainee is performing in that interval. Insights may include that the exercise is too easy for the trainee or the trainee cannot maintain the correct posture for the whole session.
3. Performing in an on-line environment: Any method that is designed to alert weightlifters for the possibility of injuries needs to be able to work while the weightlifter is performing the exercise. This is so that the alarm is raised as soon as a risk of injury is detected.

In the following chapters, we focus on taking each of these steps into a resource

limited environment of a wearable while applying them online, real-time and in-place. In Chapter 4 we present the first online, real time and unsupervised technique that runs on a wearable device. Then, in Chapter 5 we presents our novel technique for analysing time series by their shape through functional geometry. Finally, in Chapter 6 we demonstrate the ability of the techniques we designed in Chapters 4 and 5 by designing and developing the first wearable device that can detect, track and analyse weight training exercises in place and in real-time.

Chapter 4

Detecting and tracking exercises in real-time

4.1 Introduction

In this chapter we revisit the recorded exercise figure from Chapter 1, depicted in Figure 4-1. Figure 4-1 illustrates a sample data stream recorded for three different weight training exercises, performed by a trainee using a motion sensor attached to his wrist. The data is recorded in quaternion units that show rotation in space. Each exercise is delineated from the rest of the data stream using vertical bars. The repeated pattern, which shows the exercise performed by the trainee, is shown in red for each set. Comparing each exercise set, Figure 4-1 shows how much each repetition (red segments from each exercise interval) changes from one set to another in both shape and frequency. The figure shows that the shape of the data not only varies from one set to the next (as indicated by the numbered regions), but also from one repetition to the next (with different levels of amplitude). We can also see how the data corresponding to the rest periods is substantially different to the data during the exercises.

We refer to such a repeating pattern as an Interval of Recurrence (IoR). As discussed

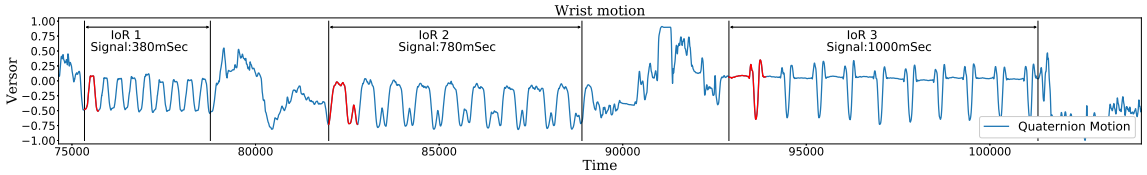


Figure 4-1: Intervals of recurrence in a time-series. Three intervals of recurrence (IoR) are shown with the recurring signals in red.

in Chapter 1, identifying these IoR in real-time on a wearable device plays a key role in developing warning systems that can detect abnormal activities as they occur, which is crucial in applications such as detecting abnormal performance by athletes to prevent injuries. The challenge with designing methods that can run on a wearable device is that they must be sufficiently efficient to run on a device with limited computing resources [5]. Thus, efficiently detecting and tracking the repeating interval (IoR) is a key requirement for designing warning systems, where a delay in detection can increase the risk of injury.

Detecting and tracking repeating patterns in real-time, without using labelled data, involves two key tasks: (1) Detecting when an IoR starts in the incoming data; (2) Tracking the consecutive repeats of the IoR while it occurs. To achieve these tasks we seek a unique characteristic of any IoR that can be computed on a wearable device to detect and track the IoR from an incoming data stream. We show that correlation is a promising operator for extracting such a characteristic, given that it has been widely used for efficiently detecting similarities in time series [139, 61, 140]. We prove that the periodic points from the autocorrelation of an IoR have a linear relationship, and use this property to design the first efficient, unsupervised and online method that can be run on a wearable device to identify and track an IoR. Table 4.1 summarises the design goals of our approach and their practical impact.

Any unsupervised machine learning method for online, real-time extraction and detection of IoR on a wearable device must address the following four challenges [141]:

1. The sparse occurrence of these activities (an IoR is often surrounded by environmental noise or other irrelevant patterns);

Table 4.1: Design goals and their impact on the system for detecting and tracking Intervals of Recurrence. The Section column indicates the section in this chapter that addresses each design goal.

G#	Design Goals	Impacts	Section
1	**Detects and tracks in real-time**	Real-time feedback	4.5
2	**Runs on a wearable**	Ubiquitous usage	4.7
3	**Performs unsupervised detection and tracking**	Adapt to new/unseen routines	4.4
4	**Adapts to variations**	Personalised	4.3
5	**Handles environmental noise**	Robust to environmental noise	4.6

2. The spatio-temporal variations in the activities (the repeating patterns can vary over time both locally and globally);
3. The limited data available from which to identify the IoR;
4. The limited computing resources available on wearable devices.

To the best of our knowledge, there is currently no state of the art algorithm that satisfies all the design goals in Table 4.1. In summary, our contributions are:

- We present an efficient, unsupervised algorithm for online detection and tracking the IoR in real-time, which satisfies the design goals in Table 4.1.
- We demonstrate the ability of our algorithm to execute on a wearable device by developing the first wearable that can detect and track IoR in a real-time manner unsupervised.
- Our method has advantages compared to current state-of-the-art algorithms in eliminating the need for intensive data collection, labelling and tuning while yielding only a small loss of accuracy (less than 3%).
- We provide in-depth theoretical analysis on how to find IoR from a continuous signal by proving the linearity and periodicity features of local maxima from auto-correlation of the signal inside an IoR.
- We provide a theoretical analysis of the impact of noise and the digitisation of the signal on our algorithm and show the robustness of this algorithm to variations in the repeating patterns.

4.2 Problem statement

Our goal is to design an efficient real-time and unsupervised algorithm that can run on a wearable device to detect Intervals of Recurrence (IoR)—the temporal regions in a time series where a short burst of consecutive repeating patterns occurs. While we focus on the application of activity recognition, this problem arises in domains such as energy management [142], where these intervals characterise electricity consumption patterns, or seismology [143], where intervals of recurrence relate to detecting incidence of earthquakes from a seismograph. We formally define our weight training activity segmentation problem in terms of identifying IoR of a time series. We define a time series as follows:

Time series T: an ordered consecutive sequence of measurements t_i: $T = \{t_1, t_2, ..., t_n\}$ where n is the length of the time series and t_i is a real valued variable measured at index i from a continuous signal, where $i \in I$ and I is an index set mapped to a set of (equidistant) time points.

Region R: an ordered consecutive sequence of measurements from T where $R \subset T$ with length m.

We consider an IoR to be a region where a recurring activity occurs, and therefore, the target of the segmentation process. In this term, *recurrence* refers to the consecutive repetitions of an activity inside the region. We model the continuity of repetitions as a periodic function that occurs for a short period of time (region) in the time series.

Interval of recurrence $R_p \subset T$ **(IoR)**: a region where there exists a periodic function $f \restriction_{R_p}$ with period K_p and a signal S such that $\forall\, t \in R_p, f \restriction_{R_p}(t + K_p) = f \restriction_{R_p}(t)$, where $\restriction$ is the restriction symbol, % is the modulo operator and

$$\forall\, t \in R_p, S(t) := \begin{cases} f \restriction_{R_p}(t) & \text{if } t < K_p \\ f \restriction_{R_p}(t \% K_p) & \text{if } t \geq K_p \end{cases} \tag{4.1}$$

We call S the Signal of Recurrence (SoR), i.e., S is the signal that repeats inside the IoR. An example of three different IoRs is shown in Figure 4-1. In each IoR, the SoR is marked in red.

Given a time series T, we want to find the set of intervals of recurrence IR = $\{R_p\}$ in T, where $\forall R_{p_j} \in$ IR an unknown integrable function $f_j \restriction_{R_{p_j}}$ (SoR) is recurring for a short period of time. The functions $f_j \restriction_{R_{p_j}}(t)$ are independent of each other and the period K_{p_j} can vary from one interval to another.

4.3 Finding IoR

In this section the goal is to seek a property of the time series data stream that can be used to detect an IoR while receiving data, online, on a wearable device. Thus, we require such a property that can be computed efficiently. To address these requirements we consider the properties of the correlation operator, which has been widely used for efficiently detecting similarities in a time series [139, 61, 140]. Our key observation is that local maxima of the autocorrelation (i.e., the correlation of a signal with itself) of an IoR from a continuous signal are (1) periodic, and (2) linearly related.

An example of the periodicity and linearity property of local maxima from the autocorrelation of an IoR is shown in Figure 4-2. In this figure, the IoR is shown in the bottom curve where a SoR is repeated three times. The autocorrelation function for that interval is drawn in the top curve. The autocorrelation is calculated by zero padding the periodic function outside the IoR. The dashed line connects all the local maxima (which occur at a period of every 20 time intervals) from the autocorrelation function, thus illustrating this linearity property. We formally present these two properties in Proposition 1.

Proposition 1. *Given time series T and an IoR $R_p \subset T$, then $\exists P=\{p_1, p_2, p_3, ...p_m\} \subset \hat{f} \restriction_{R_p}$, such that $p_1<p_2...<p_m$ are linear and periodic points where $p_i, 1 \leq i \leq m$ is a local maximum.*

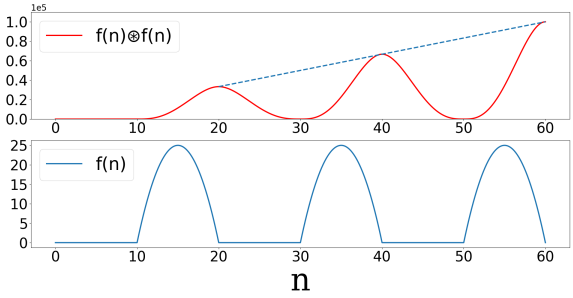


Figure 4-2: An example of a periodic function f (bottom figure) and its autocorrelation function $\hat{f}$ (top figure). The line that passes through all the maxima in $\hat{f}$ is dashed.

Proof. For readability we provide the detailed proof in Section 4.11.1. □

We use these two properties of an IoR to design an efficient algorithm to detect and track IoR online. In the Appendix (Section 4.11.1), we prove that the (1) periodicity and (2) linearity of the peaks from the autocorrelation of an IoR are general properties of any IoR, and thus can be used to detect IoR. Further, we want to make sure that these properties also hold after the digitisation of a continuous signal, and in the presence of sampling noise. In Section 4.6, we discuss the effect of sampling noise and the digitisation of the continuous signal on these two properties.

4.4 Identifying and tracking IoR online

In the previous section we proved that (1) linearity, and (2) periodicity are the two main properties of a set of local maxima from the autocorrelation of any IoR. We use these two properties to design an online algorithm for detecting and tracking IoR without the need for any labelled data, i.e., in an unsupervised manner. A flowchart of this algorithm is shown in Figure 4-3. We start by analysing the incoming raw time series data using a sliding window of size W (Figure 4-4a). At each iteration we calculate the autocorrelation of the region inside the sliding window and search for peaks that satisfy the periodicity and linearity conditions of the IoR. We test for the linearity of the peaks within a given margin of error as shown by grey lines in the figures. The peaks must fall within the error

bounds shown by the two grey lines to be considered linear. We discuss this error margin in Section 4.6. If no set of peaks satisfies the two properties, we check the current size of the window. If it is larger than a threshold (Θ), we slide the start of the window forward until the window size is equal to its initial value, W. Otherwise, we extend the window forward by the initial value W (details in Algorithm 3). If we find a region that satisfies the two properties (Figure 4-4b), we slide the window to the start of the region (step (a) in the flowchart) and extend the window by the size of the estimated period (step (b) in the flowchart, Figure 4-4c). We do this extension until no new local maximum satisfying the periodicity and linearity conditions is found through the autocorrelation of the region (Figure 4-4d). We then slide the window to the end of the IoR and continue (step (c) in the flowchart, Figure 4-4e). In the flowchart, the NewRec function returns true if a new SoR is found inside a given IoR and the NoRec function returns true if no new SoR inside the window has been seen.

The threshold Θ is defined for efficiency reasons. Setting Θ too big makes calculating the autocorrelation of the sliding window too expensive to run in real-time. Θ stops the sliding window from being extended without bound. We set Θ to be at least three times the maximum expected length of any SoR in the application domain. Since two peaks can

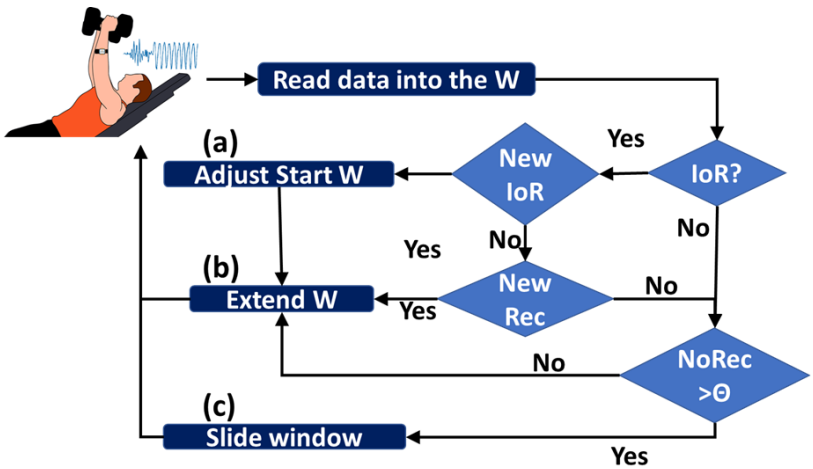
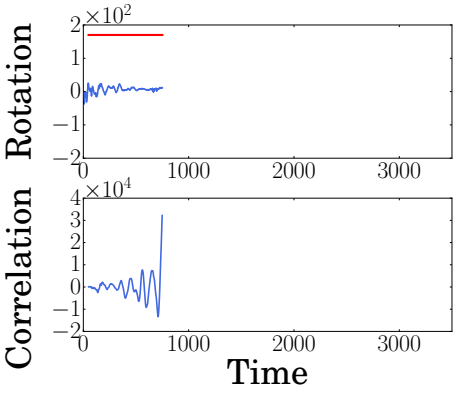
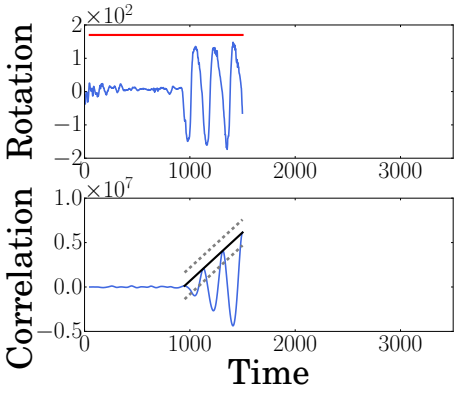


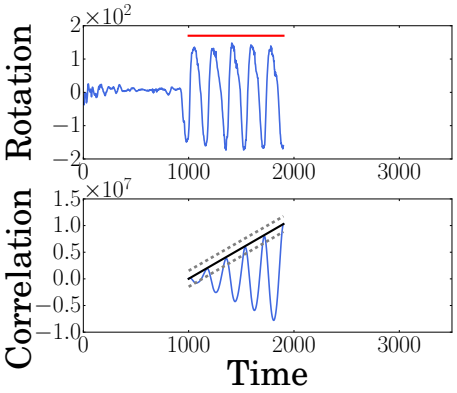
Figure 4-3: Flowchart for online tracking of IoR in real-time. W is the dynamic sliding window.



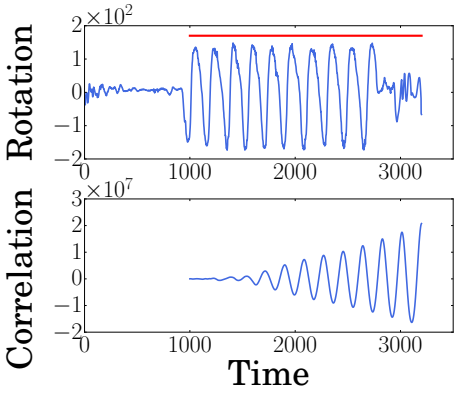
(a) No IoR.



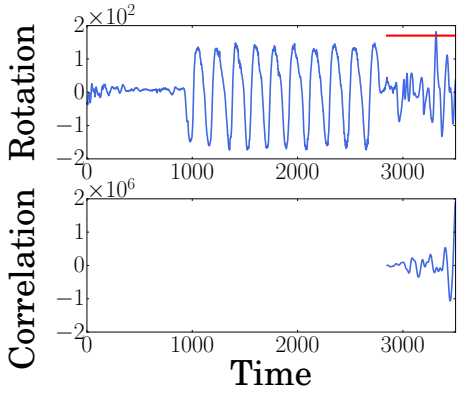
(b) Finding a possible IoR.



(c) Adjust and extend W by the estimated period.



(d) Extend W by the estimated period until no new SoR found.



(e) Slide W to the end of IoR and restart.

Figure 4-4: Online tracking of IoR in real-time. The red flat line shows the state of the sliding window. The signal in the top figures is the incoming data stream and the bottom signal is its autocorrelation of the window. The black line shows the found IoR.

always be connected using a line, our algorithm requires at least three peaks to be seen to check for the linearity condition. Thus, if we set Θ to be less than three times length of a SoR, then the sliding window will slide forward before finding the region and thus the IoR

would be ignored. However, Θ does not have any upper bound. We empirically evaluate Θ for the weight training application in more detail in Section 4.9.

The **O**nline **T**racking **o**f **R**ecurrence (OToR) algorithm for the flowchart in Figure 4-3 is shown in Algorithm 3. The algorithm uses two main parameters (1) st (start), and (2) end to create a dynamic sliding window. The aim is to grow the size of this window such that it includes the whole IoR. The algorithm accepts the initial window size W and creates a window of size W (line 1). The algorithm starts by receiving data at line 4. After receiving

Algorithm 3: Online Tracking of Recurrence (OToR)

```
Θ, Global variable
Function OToR(W):
    Data: minimum window size W
    Result: Report each found repeat from an IoR
1   st,end ← 0,W;
2   T, currentPeaks ← ∅, ∅;
3   while True do
        // Compute while receiving data
4       newData ← readData();
5       T ← T + newData;
6       while end < len(T) do
7           window ← T[st:end];
8           window ← window-mean(window);
9           C ← window ⊛ window;
10          M,MI ← MaxArgmax(C);
11          newPeaks ← FIoR(M,MI);
12          if newPeaks≠ ∅ then
13              if currentPeaks= ∅ then
14                  st ← adjust(newPeaks)
15              if len(newPeaks)≠len(currentPeaks) then
16                  currentPeaks ← newPeaks;
17                  ReportNewRepeat(newPeaks);
18              end ← end+ GetPeriod(currentPeaks);
19          if NoRec(window, currentPeaks) > Θ then
20              st ← BreakPoint(currentPeaks);
21              currentPeaks ← ∅;
22          else
23              if newPeak= ∅ then
24                  end ← end + W;
25      end
26  end
```

enough data to fill the sliding window (lines 2-3), OToR calculates the autocorrelation of the current sliding window (lines 7-9). OToR calls FIoR to check the periodicity and linearity conditions of the local maxima from the window's autocorrelation (lines 10-11). If an IoR is reported by FIoR (newPeaks is not empty, line 12), and the reported IoR is a new IoR (currentPeaks is empty), OToR slides the start of the sliding window to the start of the new IoR (line 14). If FIoR has reported a new SoR (an IoR that OToR is already tracking—currentPeak is not empty) (line 15) OToR updates the IoR and reports the new SoR (lines 16 and 17). If FIoR reports an IoR, the sliding window extends by the estimated period from the reported peaks (line 18). OToR checks the time lag from the last seen SoR at line 19. If the time lag is bigger than Θ, OToR updates the starting point (st, line 20) of the sliding window to the end of the last SoR (if no SoR observed, st updates to $end - W$). In line 23 and 24 OToR checks if no-IoR is observed yet and the sliding window is still smaller than Θ; if true, it extends the window size by W (note that if any of these two conditions were false the window has already been updated).

4.5 Multiple peaks: an efficiency problem

A challenge arises when the activity recording shows a complex signal with multiple peaks. We refer to a SoR that contains multiple peaks as a non-convex signal. Figure 4-1 shows real-world examples of non-convex signals (IoR 2 and 3). A simpler example of SoR with multiple peaks is shown in Figure 4-5 (bottom figure). The SoR is repeated three times in this figure. Taking the autocorrelation of such a SoR gives rise to multiple sets of local maxima. One set of local maxima is generated when two signals of recurrence are aligned with each other while taking the autocorrelation of the IoR. The other set of local maxima are produced when the non-dominant part of the SoR (the part with smaller peaks) matches the dominant part (the part with dominant peaks) of the SoR while taking the autocorrelation of that interval. These two sets are shown in Figure 4-5.

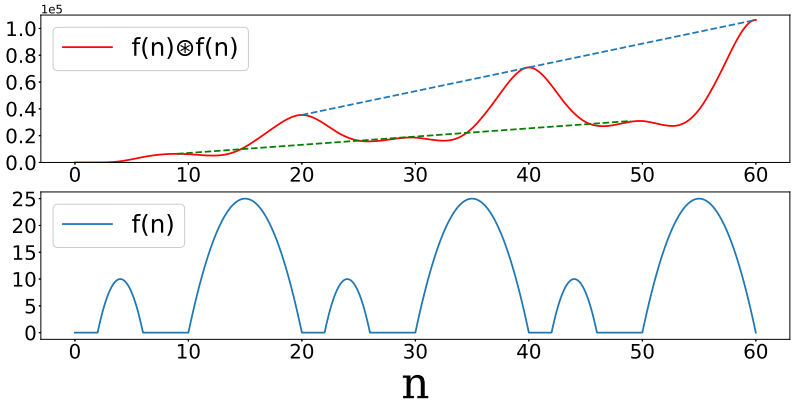


Figure 4-5: Multiple maxima from a non-convex SoR.

The maxima that are aligned with the dashed blue is the set of maxima that Proposition 1 identifies. The second set of local maxima, which complicate the detection of the IoR, are connected with a dashed green line.

A naive solution to overcome this challenge is for the FIoR function in Algorithm 3 (line 17) to perform an exhaustive search in all local maxima found by the algorithm to find a set of local maxima that satisfy the linearity and periodicity properties. However, this exhaustive search is computationally inefficient for real-time applications. Given that the goal is to run the algorithm on a wearable device with limited computing resources, we need to design an efficient algorithm that can find the subset of local maxima that satisfies the periodicity and linearity properties from a given set of local maxima.

Next we show that the linearity property of local maxima from Proposition 1 can be replaced by an arithmetic progression property, i.e., the differences between consecutive peaks' values in Proposition 1 are equal.

Proposition 2. *The set of local maxima* $L = \{l_1, l_2, \dots l_k\}$ *that satisfy Proposition 1 forms an arithmetic progression.*

Proof. Proposition 1 proves the periodicity of local maxima. The proof follows from the linear property of this set. □

Proposition 2 shows that each subset of local maxima that satisfy Proposition 1 follows

an arithmetic progression, i.e., in the set of local maxima that satisfy Proposition 1, the amount of increase from one local maxima to the next consecutive local maxima is constant for the entire set. Since the local maxima are generated with the same period, each subset of local maxima that fulfil Proposition 1 follows the same period inside L. Therefore, we can reduce the search for local maxima in L to *finding a subset in L with the same period and arithmetic progression*, which is easier to solve than searching for a subset of local maxima that are linearly aligned. In OToR we modify the FIoR function (Finding Interval of Recurrence, Algorithm 4) to search for periodic peaks with equal differences (arithmetic progression) inside the autocorrelation of the dynamic sliding window.

FIoR uses the $step$ variable to select the consecutive peaks. It starts from $step = 1$, i.e., selecting the peaks that are one after another. FIoR selects three consecutive peeks starting from the last peak in the set. If the three selected peaks do not satisfy the two conditions (periodicity and arithmetic progression), it increases $step$ by one and selects three peaks that have $step-1$ peaks in-between (the last peak in the set is always selected). If both conditions are met at any $step$, FIoR calls IoR to traverse the set of local maxima backwards and returns the subset of peaks that are $step$ distance apart with the same arithmetic progression (line 7), otherwise it increases $step$ until it reaches half of the number of peaks. FIoR returns an empty set if no set that satisfies the two conditions is found.

4.5.1 Complexity

The time complexity of the OToR algorithm depends on the time complexity of the FIoR algorithm. We define ω as the size of the biggest IoR inside the time series with a maximum number of R repeats, W as the initial size of the window, and n as the number of local maxima passed to FIoR.

In the situation where no subsequence of peaks satisfies the periodicity and linearity

Algorithm 4: Find Intervals of Recurrence

```
1  Function FIoR(M, MI):
       Data: M: Local maxima of autocorrelation of time series, MI:indices of local maxima in M
       Result: R = {(s,e)|R_p = [s,e]}
2      step ← 1;
3      while step < len(M)/2 do
4          i ← len(M);
5          P, PI ← M[i,i-step,i-2*step], MI[i,i-step,i-2*step];
           // the last 3 items with 'step' distance apart from the inputs
6          if PERIODIC(PI) and AP(P) then
7              return IOR(PI, P, step);
8          step ← step+1
9      end
10     return ∅;
```

conditions, FIoR traverses the local maxima once. FIoR selects the last three peaks with 'step' distance apart (Line 5), checks for the two conditions (Line 6) and increases the 'step' variable until all the subsequences fail the two conditions (Lines 3). Each check for the two conditions is of constant time (the PERIODIC procedure checks if the peaks' indexes are the same distance apart and AP checks if the peaks' values are the same distance apart). In case there exists a set of consecutive peaks that satisfy arithmetic progression and periodicity conditions (Line 6), then FIoR uses the IOR subroutine to find all periodic peaks with the same estimated arithmetic progression and period. IOR needs to traverse the list of local maxima once to find all the peaks with the given estimated period and arithmetic progression, which is $O(n)$ in time, where $n = |M|$ and $|M|$ denotes the length of the set M. Therefore, FIoR's time complexity is $O(n)$.

The other subroutines in OToR (len, NOREC, MAXARGMAX and GETPERIOD) also have linear time complexity. The autocorrelation in line 9 uses the convolution theorem and fast Fourier transform, which has $O(m \log m)$ time complexity, where m is the input size. We define the number of times OToR extends the window K, thus, OToR runs in $O(K(n + m \log m))$. Since the FIoR input parameters are the local maxima computed from the autocorrelation of the dynamic window together with their locations, FIoR's maximum input size, n, is equal to $m/2$. Thus, OToR's time complexity is $O(Km \log m)$. To find the

maximum values for K and m we need to know how the dynamic window size changes. The dynamic window has three stages: (1) The dynamic window extends in a region with no IoR. In this situation the maximum size of the dynamic window is Θ (the threshold explained in Section 4.4); $m = \Theta$. The window size extends by W until its size is Θ; $K = \Theta/W$. As a result, in this scenario, OToR runs in $O\big(\Theta/W(\Theta\log\Theta)\big) = O\big((\Theta^2/W)\log\Theta\big)$. (2) The dynamic window has found an IoR. In this case, the window extends by the estimated period, i.e., it extends R times ($K = R$). The maximum size of the window is ω; $m = \omega$. In this scenario, OToR runs in $O(R\omega\log\omega)$. (3) The IoR tracking is finished and OToR extends the dynamic window until it reaches the threshold Θ. Thus, $m = \omega + \Theta$. At each iteration the window is extended by the estimated period p, thus, $K = R + \Theta/p$. In this scenario, OToR runs in $O\Big((R+\Theta/p)\big((\omega+\Theta)\log(\omega+\Theta)\big)\Big)$.

In total, OToR runs in $O\Big((R+\Theta/p)\big((\omega+\Theta)\log(\omega+\Theta)\big)+R\omega\log\omega+\Theta^2/W\log\Theta\Big) = O\Big((R+\Theta/p)(\omega+\Theta)\log(\omega+\Theta)+\Theta^2/W\log\Theta\Big)$. Since Θ is a constant that is defined at the start, OToR runs in $O(R\omega\log\omega)$. Note that R is often a small value (for example in weight training it is a number between 6-15). This complexity enables us to run the algorithm on computationally constrained devices in real-time, as we demonstrate in Section 4.7. We investigate the effect of Θ on OToR in Section 4.9.

4.6 The effect of noise

Proposition 1 shows how to identify and track IoR in a continuous system. However, in reality we digitise the signals for finding and tracking IoR. In this section we model the effect of digitisation and noise on the algorithm. According to Nyquist's theorem, if a signal is sampled at more than twice the highest frequency presented in its Fourier domain, the signal can be reconstructed without error [144]. The discrete version of Proposition 1 for finding IoR using autocorrelation can be written in summation form over the sampled signal. If the IoR length is equal to N and $f(m)$ is the discrete function,

we can rewrite the autocorrelation function as: $\hat{f}(n) = \sum_{m=0}^{N-1} f(m) f(n-m)$. Hence, in practice three important factors can affect this calculation, namely:

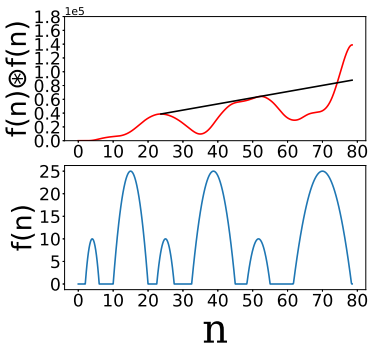
1. **Jitter** results from hardware clock limitations in reproducing the true periodic signal. In an ideal scenario, we would be able to sample the signal in exact τ intervals. However, jitter perturbates the sampling points.
2. **Amplitude noise** is the small error present around each read of the sampled signal. This noise often results from two sources: (a) read errors, and (b) performance errors. The read error is an inseparable part of any digitised device where each read can be affected by environmental factors such as temperature or humidity. The second source of amplitude error comes from the signal's source where repeats cannot be performed 100% identically.
3. **Discretization error** shows itself as variations in repetitions of the digitised SoR in an IoR. A sampling rate that is not synchronised to the period of repeat in an IoR introduces discretization error. In this case, since the period of repeat in the IoR is not factorable to the sampling rate, samples are taken at different points from the successive repeats of the waveform. Thus, each digitised SoR from the IoR varies from one repeat to another even though all the original SoRs are identical in the IoR.

A complete analysis of each type of noise is provided in Section 4.11.1. In a nutshell, in scenarios where the noise is known with a bounded variance or is modelled by a Gaussian distribution, Proposition 1 still holds, i.e., the expected line in the presence of Gaussian noise calculated from a discretized signal is the same as the line from Proposition 1 with a bounded variance (for the detailed proof see the Section 4.11.2). The variance is an important factor in OToR. Earlier in Section 4.4, we discussed setting a marginal error for the linearity property of the local maxima. The marginal error is directly set from the variance for the linearity property of the local maxima. In Section 4.9 we empirically

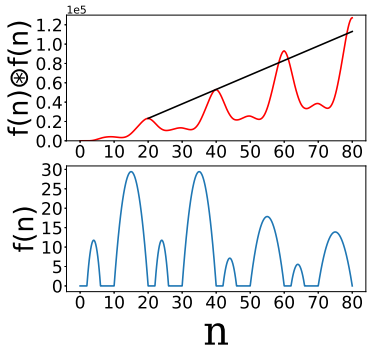
demonstrate the effect of this parameter on OToR. In this section, we also discuss non-Gaussian noise and how it can be handled.

There are situations where there is systematic noise in the period of recurrence. In these cases, the SoR period gradually increases (decreases) over time. Figure 4-6a shows an example of the period of an SoR increasing over time. Another scenario is for the amplitude of an SoR to gradually decrease (increase) over time, as shown in Figure 4-6b (e.g., a trainee becomes too fatigued to maintain the correct form). In both situations the relations between local maxima are not linear, which stops OToR from detecting these IoR. Thus, if OToR finds an IoR, it shows that the SoR inside the IoR has been consistent, which is a desirable feature for providing feedback. For example, if OToR detects an entire exercise set it tells the trainee that he/she has maintained the exercise routine for an entire set.

An example of a trainee failing to maintain the routine is shown in Figure 4-7. In these situations, the slope of the line from Proposition 1 changes gradually. As shown in the figure, the slope of the line changes from the first three repeats to the last three. We propose that changes of the slope of the line from Proposition 1 can be an accurate indicator of the presence of systematic noise in the IoR, for example, the trainee is becoming fatigued. Analysing this effect is beyond the scope of this thesis and left for



(a) Systematic changes of the period of SoR inside an IoR.



(b) Systematic changes of the amplitude of SoR inside an IoR.

Figure 4-6: Systematic error results in changes of SoR in the IoR, which breaks the linearity property of local maxima from the autocorrelation of the IoR

future work.

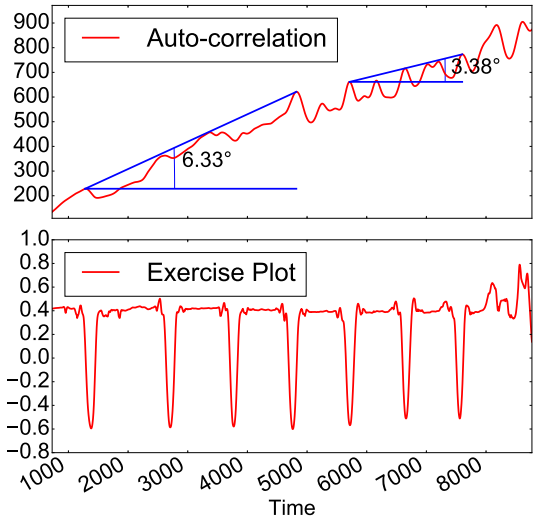


Figure 4-7: The changes in the slope of the line from IoR can show an inconsistency in the performance. The region where the trainee's performance was inconsistent is highlighted.

4.7 Implementation of OToR on a wearable device

We have implemented OToR on a wearable device and used it to monitor fitness training. The users can wear the device to the gym to automatically log their weightlifting routines. Our wearable is based on the Intel Edison platform, and SparkFun base kit blocks (a base block, a 9 DoF block and a Li-ion 400 mAh battery) to assemble the device, shown in Figure 4-8.

The Intel Edison processing unit has a dual core 500MHz CPU with 1GB of RAM. We

(a) The assembled device

(b) Wearing the prototype

Figure 4-8: The designed wearable device.

deployed the OToR algorithm using Python 2.7.14 onto the device. The Python service runs in the background, continually reading from the IMU sensor and using OToR to segment activities. The thread has a 0.5-second window to collect the data before applying the algorithm. The sensor reads at 45Hz through a separate thread and passes the data to the OToR thread. This dual thread design was selected to ensure the data is always read without interruptions.

4.8 Experimental evaluation

We empirically evaluated OToR in two scenarios: (A) Offline: running the algorithm on four datasets, and (B) Online: running the algorithm on the wearable device described in Section 4.7 in a real-world environment.

4.8.1 Offline tracking

The aim of the first scenario is to show the accuracy of OToR in comparison to the state-of-the-art alternative algorithms in segmenting the data. To perform this test, we used four different datasets, both synthetic and real data collected from users.

Synthetic idealised data (I-SYN) This dataset is an idealised scenario where the SoR remains unchanged during the IoR. We focus on how well each algorithm performs on a perfectly consistent dataset. This scenario provides a baseline to analyse the behaviour of each algorithm after introducing variations of the SoR into the dataset. To generate this dataset, we used two polynomial functions: (a) a second degree polynomial function $S(x) = ax^2 + bx + c$; and (b) a third degree polynomial function $S(x) = ax^3 + bx^2 + cx + d$, as the SoR. For each SoR, S, the function's domain is set to be a short period around its root—$S(x) = 0$. The signal is repeated for a random number of times: $\forall t \in R_p, f\restriction_{R_p}(t) = S(t\%K_p)$. For any other points in the time series we used a random number from a Gaussian distribution. An example is shown in Figure 4-9 (Top, Left).

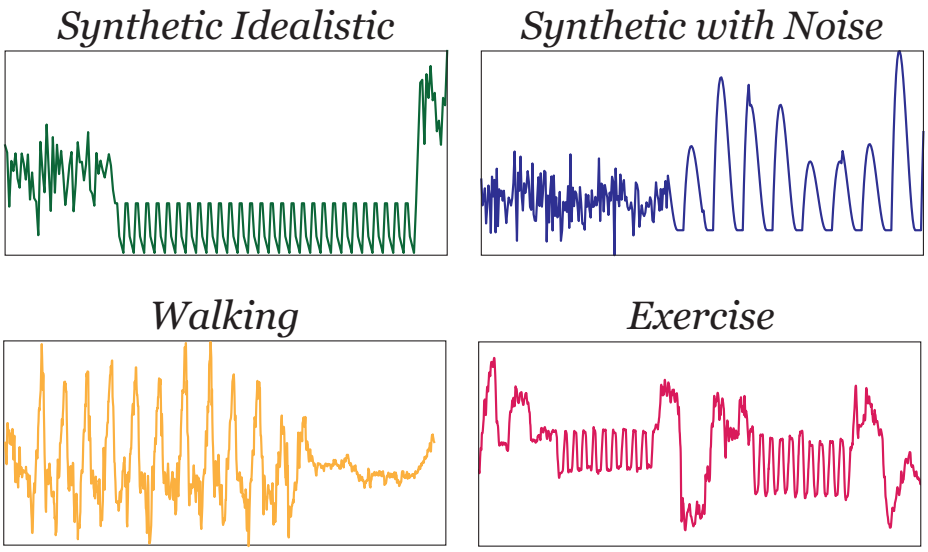


Figure 4-9: Example time series in the datasets: (a) Synthetic idealised dataset; (b) Synthetic data with variations; (c) HAR dataset; and (d) Weight lifting dataset.

Synthetic data with variation (P-SYN) A more realistic dataset has variations in each recurrence of S. In the P-SYN dataset we focus on how each algorithm reacts to variations of the SoR inside the IoR. We use the dataset from Section 4.8.1 and add a divergence factor, a random number d from a uniform distribution $U(0.5, 1.5)$, to each repetition of signal S inside the IoR, i.e., each repeat is $d * S(x)$ (Figure 4-9, top right).

Real data from on-body sensors (EXER) We used one set of six on-body motion sensors to collect the body movements of trainees during a weight training session. Each sensor was attached securely to different parts of the trainee's body: left wrist, left upper arm, lower back (lumbar), left thigh, left ankle and waist. We chose only one side of the body because weight training exercises are often symmetric movements. Thus, with a limited number of sensors we could record movements from different body parts. To make sure we can capture every possible exercise, we have to record full body movements. To report the result, we run each algorithm on the data from each sensor and then calculate the union of the results from each algorithm to find all repeating segments.

We selected 11 exercises that provide a complete range of body motions: (1) Squat, (2) Military press, (3) Front squat, (4) Dead lift, (5) Up right barbel row, (6) Sumo high pull,

(7) Side lateral to front, (8) Renegade triceps kick back, (9) Hammer curl, (10) Around the world, and (11) Dumbbell fly.

We asked two participants to perform each exercise for 10 repeats. Each session started with a personal trainer showing the participant how to perform the exercise, next the participant watched a video showing how to perform the exercise, and then performed the exercises. Participants were free to rest as required. We recorded the whole session using a camera to establish the ground truth. An example of a training session dataset can be seen in Figure 4-9 (Bottom, Right).

Human Activity Recognition (HAR) To show that OToR is applicable to finding other short bursts of repeating patterns in HAR, we used a public dataset published by Reyes-Ortiz et al. [145] (Available at `https://bit.ly/2vLvTCy`). This dataset contains continuous recordings of accelerometer data for users performing daily activities: Walking, Walking Upstairs, Walking Downstairs, Sitting, Standing, Laying, Stand to Sit, Sit to Stand, Sit to Lie, Lie to Sit, Stand to Lie, and Lie to Stand. Our task is to find regions where the user is performing a repetitive task: Walking, Walking Upstairs, and Walking Downstairs. We select regions from each participant's accelerometer data, where the users performed each of the repetitive tasks at least once. These regions include the part where the participant performed the task as well as the transitional intervals where the participant prepared to perform the next task. We used the labels assigned to the data by the authors as the ground truth and selected the first 10 users from the dataset. An example of this dataset is given in Figure 4-9 (Bottom, Left).

Baseline segmentation

To compare the OToR approach against the current state-of-the-art, we evaluated it against three other methods: MP by Yeh et al. [92], incorporating the modification introduced by Mirmomeni et al. [94], AutoPlait by Matsubara et al. [80], and DecomposeTS by Zhao and Itii [81].

Matrix Profile (MP) MP is a vector such that at each point on the time series, it stores the Euclidean distance between the subsequence of length m starting from that point, to its nearest neighbour. MP searches for subsequences of fixed length m, and aims to find the closest subsequence to each subsequence of length m [92]. MP drops to zero during an IoR. Thus, to find intervals of recurrence in a time series we can search for points on the MP with values smaller than a threshold for an interval. The main problem is that MP depends on its input parameter m to find the IoR. Mirmomeni et al. [94] showed that m must be equal to the length of the shortest non-repeating subsequence of the SoR inside an IoR to find the IoR accurately. We embed their method into MP to find IoR.

AutoPlait (AP) AutoPlait is based on the concept of regimes that the authors introduced into time series [80]. A regime is a hidden state that describes the behaviour of a segment(s) of a time series. A time series is generated by a Hidden Markov Model (HMM) of regimes that switches between/within them. To find the optimum set of regimes, the authors adopt the idea of compactness from coding theory. The goal is to generate the minimum number of regimes that best describe the HMM that creates the time series. To achieve this goal an iterative approach is proposed, which at each iteration breaks the currently found regimes into two regimes. The iteration process is governed by an objective function that describes each regime through coding theory (number of bits used to code the time series, number of segments in each regime, length of segments, etc). We extract IoR segments from the segments generated by this method using a length threshold, i.e., we extract segments whose length are within a given range.

DecomposeTS (DT) DT is based on the definition of homogeneity for a time series. The homogeneity of a time series is defined through its symbolic representation of the time series [81]. The authors used Symbolic Aggregate Representation (SAX) [82] of a time series to transform a given time series into a sequence of symbols. The authors define the entropy of a time series by the total entropy of the symbols generating its SAX string. This string of symbols is then split into two substrings if the total entropy of the new pair of

Table 4.2: Segmentation results from OToR in comparison with other methods for the four datasets described in Section 4.8.

	Precision				**Recall**				**F1-Score**			
	OToR	DT	AP	MP	OToR	DT	AP	MP	OToR	DT	AP	MP
I-SYN	**.95**	.45	.55	**.95**	.91	**1**	**1**	.91	**.93**	.62	.71	**.93**
P-SYN	**.94**	.76	.54	.74	.88	.98	**1**	.74	**.91**	.84	.70	.74
EXER	**.71**	.60	.41	.66	.76	.27	**.95**	.76	**.74**	.35	.56	.71
HAR	.81	**.88**	.36	.62	.75	.53	**1**	.61	**.78**	.64	.53	.58

substrings is less than their parent. The dividing process stops when the decomposed tree is at its local minimum entropy. We use this method by filtering the generated substring using an upper and lower bound threshold over the length of generated segments to extract IoRs from the time series.

Results

To compare the results obtained from the three algorithms mentioned above, we use the classification by data point scheme suggested by Lin et al. [70]. A true positive (TP) is defined to be a point where the ground truth and the result from the algorithm match with each other, and a false positive (FP) to be a point where the result from the algorithm and the ground truth does not match. A false negative (FN) and a true negative (TN) are defined similarly against the ground truth. We then calculate Precision = $(\frac{TP}{TP+FP})$, Recall = $\frac{TP}{TP+FN}$ and F1-Score = $2\frac{\text{Precision}*\text{Recall}}{Precision+Recall}$. The results are shown in Table 4.2, with bold indicating when the corresponding algorithm is the winner.

Discussion

Table 4.2 shows that OToR outperforms the other methods in all datasets in terms of the F1-Score. The main strength of the OToR algorithm results from the principle of autocorrelation, which amplifies the dominant frequency of a signal in comparison to the noise, and reduces the effect of variations. Wachman et al. [122] showed that cross-

correlation can be considered as a similarity function that focuses on the similarity of the shape of two signals. The MP algorithm, on the other hand, is affected by the presence of perturbations in the SoR, because the most similar signal to a signal inside an IoR may not be exactly consecutive to that signal. As a result, MP deviates from a flat line, and the profile shows disturbance inside the IoR. Thus MP is susceptible to changes of SoR inside the IoR. The second problem that affects MP in finding an IoR accurately is that MP is sensitive to its key input parameter M to return the best result. However, as shown by Mirmomeni et al. [94] for a given time series with multiple IoR we need to set different M to find each IoR accurately, thus a single M value cannot find all the IoR.

The situation for AutoPlait and DecomposeTS is different from MP. These two methods are based on statistical features from the time series to find the optimum location to segment (split) the time series. This characteristic causes two general issues: (1) Failure in situations where the data is not sufficient to distinguish between statistical changes, and (2) Failure to differentiate between IoR with similar but not identical distributions. The first problem arises when the IoR is very short compared to the rest of the time series. For example, in a 90 minute gym session an exercise set might take up to 60 seconds, which is less than one percent of the whole session. Thus the two methods are highly likely to ignore these short bursts of repeating patterns. For example, consider the two IoR segments that can be seen as the last two red segments in the ground truth (GT) section of Figure 4-10. DT completely missidentifies these two IoR segments because their lengths are too short to provide enough data for the algorithm to detect the regions (Figure 4-10 DT section).

The second problem for AP and DT shows itself when two IoR take place within a short distance of each other (for example, the first three IoR in the GT in Figure 4-10). In this situation, both algorithms tend to consider the two regions as one, and segment the two IoR together. The corresponding long red segments in DT and AP in Figure 4-10 demonstrate this problem. The high recall and low precision results from DT and AP in

Table 4.2 also confirm this problem.

The high precision and recall (over 0.9) together with high F1-Score (.95) of OToR from synthetic noiseless dataset demonstrate that linearity and periodicity are the two distinctive properties of an IoR. In this scenario, the segments where OToR is most likely to fail are at the very beginning of the IoR. This region may fall into the last part of the sliding window and thus the window can break one or two repetitions of the signal of recurrence from the IoR. As a result the sliding window ignores the first couple of repeats at the start of the IoR by sliding to the third repetition.

These results demonstrate the robustness to variations in the SoR by correctly identifying intervals of recurrence in the P-SYN dataset. As seen in the results, a small change in the SoR does not impact the ability of the OToR algorithm to detect intervals of recurrence. In this scenario, OToR's ability to correctly identify intervals of recurrence only drops by 1%. Although recall remains close to 0.9, OToR appears to be more sensitive to variations in the SoR. This becomes an issue when the end part of an IoR is entirely located inside the sliding window. If the amplitude of the last signals of recurrence degrade, the transition from IoR to non-IoR becomes blurred, i.e., we find peaks from the autocorrelation of the transitional region that are not part of the IoR.

Further, we tested OToR using two real world datasets (weight training—EXER, and human activity recognition— HAR [145]). In both cases, OToR outperformed the state-of-the-art. OToR had a higher precision in the HAR dataset compared to the EXER because activities in this dataset are much easier to perform compared to the EXER dataset. In EXER dataset, the very last repetitions may deteriorate substantially over time as shown by Melchiorri and Rainoldi [146], which causes the transition from IoR to non-IoR to be harder to distinguish.

4.8.2 Online tracking (counting repeats)

We evaluate the effectiveness of OToR to detect and count repetitions online in a real-life environment (a gym).

Personal Trainers Dataset (PTD) We recruited seven professional personal trainers to perform four different exercises in the gym using the wearable device described in Section 4.7. The wearable device was worn on the right wrist of the participants throughout the trials. Each personal trainer was asked to perform two sets with ten repeats of the four different exercises: (1) Side lateral raise, (2) Bicep curls, (3) Bicep curls to shoulder press, and (4) Hammer curls.

RecoFit Morris et al. [6] designed a wearable that uses a supervised approach to count the repeats of predefined weight training exercises online. However, their data and implementation are not publicly available, making it impossible for a fair and accurate comparison. The authors have not provided any detail about the performance of their algorithm in tracking recurrences online. We contacted the authors to gain access to their data and/or method but they revealed to us that both are private and not available for public use.

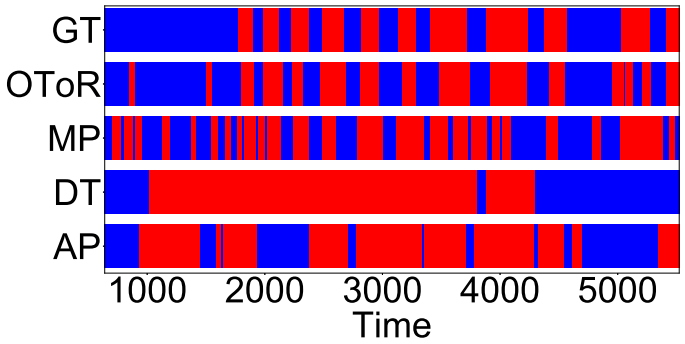


Figure 4-10: Algorithms' results in segmenting a real gym session. Regions in red correspond to inferred IoR (GT ground truth).

Results

We use the same technique by RecoFit to compare their reported results from their training set to OToR's online results for tracking of exercises [6]. RecoFit has reported their exercise counting percentage from their training sets (Table 4.3). In this table, we show the average accuracy of counting repeats for the four exercises. OToR can achieve high accuracy with a margin of two mistakes, while completely eliminating the need for preparing a large training dataset. Figure 4-11a shows the distribution of counting errors per exercise in our experiment.

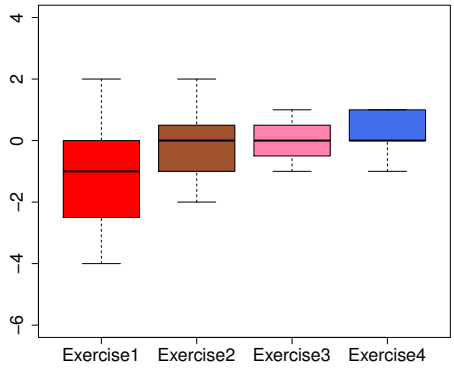
Table 4.3: Online tracking. Exact: the percentage of training sets that were counted precisely. Within-1: the percentage of training sets that had an error of up to 1 count. Within-2: the percentage of training sets that had an error of up to 2 counts. NA means the result is not available.

	Exact	**Within-1**	**Within-2**	**Proc(Sec)**	**Exec(Sec)**
OToR	47%	81%	96%	0.09	1.5
RecoFit	70%	93%	97%	NA	NA

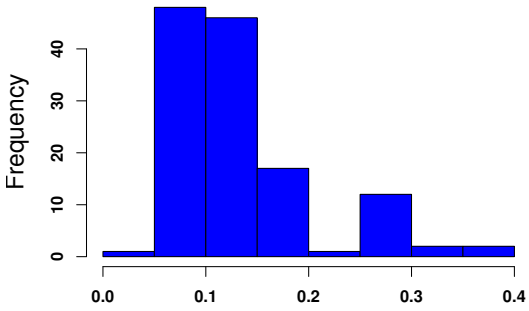
Conclusion

Although, as shown in Table 4.3, OToR's performance in counting exact repeats is lower than *RecoFit*, our algorithm does not require any supervised training, bootstrapping, or prior data. In contrast RecoFit is a supervised learning method, which requires an expert to watch the participants while they perform the exercises to label the dataset. To detect 13 exercises, the authors of RecoFit went through 126 sessions of data collection before being able to train their classifier. To account for personal differences they collected data from participants with various ages, genders and abilities. All these efforts show the importance of our work in unsupervised tracking of exercises. Note that since OToR detects the SoR, a post processing algorithm that finds each SoR from the IoR will improve the counting result.

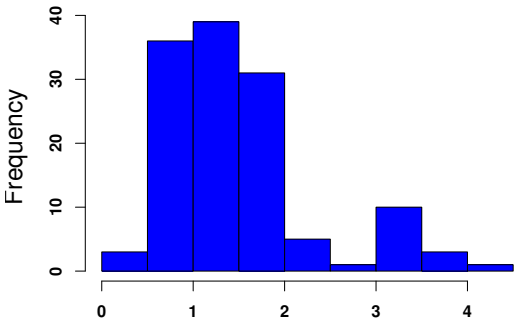
Table 4.3 shows that OToR algorithm can detect each repeat from an online sequence



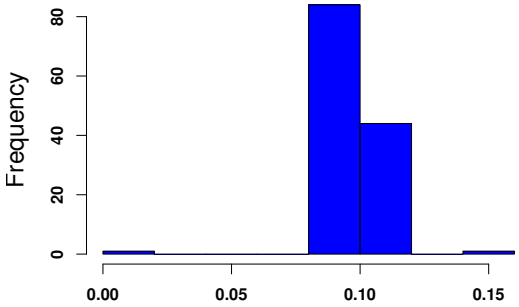
(a) Error count for each exercise. Zero means exact count.



(b) CPU time-lag for detecting each new repeat (in seconds).



(c) WallClock time-lag for detecting each new repeat (in seconds).



(d) Percentage of processing time for OToR algorithm.

Figure 4-11: Performance of OToR in real-time.

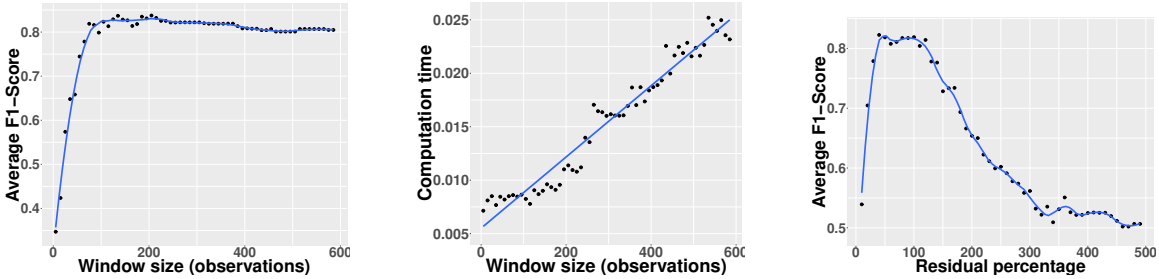
of exercises within 0.09 seconds on average (Proc column in Table 4.3). The distribution of this measurement is shown in Figure 4-11b. The wearable device's sensor read frequency was set to 45Hz. Thus, we analysed 45 reads per-second. The longest exercise was the Bicep to shoulder press, which takes around 5 seconds to perform. Each participant performed 10 repetitions of this exercise. OToR waits for 3 periods before stopping to track an IoR. Thus, in total there were $45 * (5 * (10 + 3)) \approx 3000$ reads for OToR to analyse during the 0.09 seconds. The algorithm has to wait for the raw data to be ready (read from the sensor every half a second and passed to the processor) before it can analyse

the data. Thus the execution time (WallClock in Figure 4-11c), which includes the time for in-between processing and notifying the OToR for the readiness of new raw data takes on average 1.5 seconds (Exec column in Table 4.3). Typically, gym exercises take around 2 to 5 seconds per repetition. Achieving a 1.5 second lag on average allows the algorithm to analyse and provide feedback to the user in near-real time.

Figure 4-11d shows that OToR requires less than 10% of the processing time, which confirms that OToR places a relatively light load on the CPU, on average taking 9% of the cycles. In the weight training application, the two main sources of power consumption are processing time and sensor reads, therefore OToR is highly efficient in power consumption, and keeps the CPU in idle mode 91% of time for other tasks or to save power.

4.9 OToR parameters

In Section 4.4, we discussed that OToR requires the configuration of two parameters—the window size threshold that stops the dynamic window from extending without bound and the marginal error threshold for linearity to track variations of SoR.



(a) The effect of window size threshold (Θ) on accuracy.

(b) The effect of window size threshold (Θ) on computation time.

(c) The effect of marginal error threshold on accuracy.

Figure 4-12: The effect of OToR parameters on its performance.

1. Window size threshold Θ: The threshold has only a small effect on the performance of the algorithm, if set within a reasonable range. To show this, we evaluated the effect of Θ

on the F1-Score of the segmentation task using the PTD dataset described in Section 4.8.2. We ran the experiment for all seven participants using a range of window sizes from 5 (0.1 second) to 600 observations (13 seconds). We calculated the average F1-Score for each window size, averaged across all participants. The results are shown in Figure 4-12a. This figure shows that the F1-Score increases until it plateaus at around a 100-sample window, with little change as the window size increases. Below 100-samples, there is not enough data to capture the three necessary repetitions for the system to start counting, but if the threshold is large enough to capture the three repeats, there is no further gain in setting a larger threshold. In contrast, Figure 4-12b shows that the time required to process the window increases linearly by increasing the threshold. This suggests a trade-off between the robustness of the system (for which the marginal gains diminish after reaching a threshold) and the responsiveness of the system (which is progressively reduced as we increase Θ).

2. Marginal error threshold: This is the threshold for determining how close a newly observed peak should be from the arithmetic progression calculated using previously observed peaks. To understand the effect of this threshold on the system's performance, we calculated the F1-Score using a range of thresholds on the PTD dataset described in Section 4.8.2. The result of this experiment, averaged across all participants, is shown in Figure 4-12c. The best performance was achieved with a marginal error threshold of 40 to 120%. If the threshold is too small, it makes the recognition too strict, forcing the user to be unnaturally consistent in performance. If the threshold is too large, it decreases the robustness of the system, accepting movements that do not consist of a new repetition of the same exercise. Within the 40-120% range, the recognition remains stable, giving a wide margin of error for setting this parameter.

4.10 Conclusion

In this chapter we designed OToR, a novel and efficient algorithm to find and track intervals of recurrence (IoR) from time series data streams in real-time on a wearable device. We demonstrated the robustness of this algorithm, which is built on (1) periodicity and (2) sub-domain linearity properties of autocorrelation. We provided an in-depth theoretical analysis of these principles.

We achieved two design objectives that go beyond the state-of-the-art. First, OToR tracks the activity performance with minimal prior knowledge and configuration, i.e., unsupervised. Second, the tracking happens in a device with low computing/memory resources (a wearable) in real-time. We demonstrated the superiority of OToR in our experiments in both offline and online scenarios in comparison to the state-of-the-art algorithms.

4.11 Appendix

4.11.1 Linearity property of periodic points from the autocorrelation of a continuous IoR

In this section, our aim is to formally prove the two main properties that we use of an IoR—(1) linearity and (2) periodicity of the local maxima of the autocorrelation of an IoR—the two properties we discussed in the previous section to detect and track an IoR from a streaming data. In Theorem 1, we first prove that all periodic points from the autocorrelation of an IoR inside a continuous signal are linearly aligned. Next we prove in Proposition 4 that the local maxima of the autocorrelation of an IoR satisfy the periodicity of the points from the IoR and are thus linearly related. These properties are used as the bases of our algorithm in Section 4.4.

We define $\hat{f}$ to be the autocorrelation function, i.e., $\hat{f}(x) = f \circledast f$ where $\circledast$ is the

correlation operator. The autocorrelation operator for a restricted function $f\restriction_{R_p}$ (where the domain of f is R_p) is defined by applying autocorrelation after zero padding the function for the regions outside the restriction region $\forall x \in R_p | \hat{f}(x) = \int_0^x f\restriction_{R_p}(x+\tau) f\restriction_{R_p}(\tau)\, d\tau$. To prove Theorem 1, we first introduce Lemma 1 for any IoR R_p to show that for an IoR the derivative of the autocorrelation of $f\restriction_{R_p}$ follows a periodic function. Using Lemma 1 we prove Theorem 1 that if the derivative of an autocorrelation function is periodic, then the periodic points on the autocorrelation result from an addition of a linear function and a periodic function. We then prove in Proposition 3 that there exists a set of local maxima from an IoR that are periodic. Finally, we prove the linearity and periodicity of local maxima (Proposition 4) to formally show that our algorithm holds for any IoR.

Lemma 1. Given a time series T and IoR $R_p \subset T$, there exists a function $\Gamma(x)$ such that $g'(x) = \dfrac{d}{dx} \dfrac{\hat{f}\restriction_{R_p}(x)}{\Gamma(x)}$ is periodic.

Proof. The goal is to find the function $\Gamma(x)$ such that dividing the autocorrelation of $f\restriction_{R_p}$ by $\Gamma(x)$ results in a function with a periodic derivative, i.e., $\dfrac{d}{dx} \dfrac{\hat{f}\restriction_{R_p}(x+K_p)}{\Gamma(x+K_p)} = \dfrac{d}{dx} \dfrac{\hat{f}\restriction_{R_p}(x)}{\Gamma(x)}$. To find $\Gamma(x)$, we start by analysing the result of $\hat{f}\restriction_{R_p}(x+K_p) - \hat{f}\restriction_{R_p}(x)$ to check for periodicity as follows:

$$\begin{aligned} \hat{f}\restriction_{R_p}(x+K_p) - \hat{f}\restriction_{R_p}(x) = \int_0^x f\restriction_{R_p}(x+K_p+\tau) f\restriction_{R_p}(\tau)\, d\tau + \\ \int_x^{x+K_p} f\restriction_{R_p}(x+K_p+\tau) f\restriction_{R_p}(\tau)\, d\tau - \int_0^x f\restriction_{R_p}(x+\tau) f\restriction_{R_p}(\tau)\, d\tau \end{aligned} \tag{4.2}$$

Because $f\restriction_{R_p}$ is a periodic function with period K_p then $f\restriction_{R_p}(x+K_p+\tau) = f\restriction_{R_p}(x+\tau)$, thus the first and third terms from Equation 4.2 eliminate each other and we have

$$\hat{f}\restriction_{R_p}(x+K_p) - \hat{f}\restriction_{R_p}(x) = \int_x^{x+K_p} f\restriction_{R_p}(x+K_p+\tau) f\restriction_{R_p}(\tau)\, d\tau \tag{4.3}$$

Since K_p is the period of f in R_p thus we have

$$\int_{x}^{x+K_p} f\restriction_{R_p}\left(x+K_p+\tau\right) f\restriction_{R_p}(\tau)\,d\tau = \int_{x\%K_p}^{K_p} f\restriction_{R_p}(x+\tau)\, f\restriction_{R_p}(\tau)\,d\tau \tag{4.4}$$

To simplify our notation we call $\int_{x\%K_p}^{K_p} f\restriction_{R_p}(x+\tau)\, f\restriction_{R_p}(\tau)\,d\tau,\ I\left(x\%K_p\right)$. The derivative of a function is periodic if the differences between periodic values of that function are a constant. Dividing both sides of Equation 4.4 by $I(x\%K_p)$ makes the right side of Equation 4.4 a constant and as a result its derivative becomes zero, i.e., $\frac{d}{dx}\frac{\hat{f}\restriction_{R_p}\left(x+K_p\right)}{I\left(x\%K_p\right)} - \frac{d}{dx}\frac{\hat{f}\restriction_{R_p}(x)}{I\left(x\%K_p\right)} = 0$. Thus, $\frac{d}{dx}\frac{\hat{f}\restriction_{R_p}\left(x+K_p\right)}{I\left(x\%K_p\right)} = \frac{d}{dx}\frac{\hat{f}\restriction_{R_p}(x)}{I\left(x\%K_p\right)}$.

Because $I\left(x\%K_p\right) = I\left(\left(x+K_p\right)\%K_p\right)$, and defining $g(x)=\frac{\hat{f}\restriction_{R_p}(x)}{I\left(x\%K_p\right)}$ we can write

$$\frac{d}{dx}\frac{\hat{f}\restriction_{R_p}\left(x+K_p\right)}{I\left(\left(x+K_p\right)\%K_p\right)} = \frac{d}{dx}\frac{\hat{f}\restriction_{R_p}(x)}{I\left(x\%K_p\right)} \Rightarrow \frac{dg\left(x+K_p\right)}{dx} = \frac{dg(x)}{dx}$$

that is, the derivative of g, is periodic with period K_p. By defining $\Gamma(x) := I(x\%K_p)$ the proof is complete. □

The next theorem shows that for any set $r = \{t_1, \ldots t_l\} \subset R_p$, where each point in r is K_p apart from its subsequent point, there exists a line that connects all the points from $\hat{f}\left(t_j\right) | t_j \in r$.

Theorem 1. *If R_p is an IoR from a given time series T with period K_p, then for any set $Z=\{\hat{f}\restriction_{R_p}(t_1), \hat{f}\restriction_{R_p}(t_2), \ldots, \hat{f}\restriction_{R_p}(t_i), \ldots, \hat{f}\restriction_{R_p}(t_r)\}$ in $\hat{f}\restriction_{R_p}$ where $t_{i+1} - t_i = K_p$, there exists a line $z(t) = at + c$ that passes through all the points in Z.*

Proof. In Lemma 1 we showed that $g'(x) = \frac{d}{dx}\frac{\hat{f}\restriction_{R_p}(x)}{I\left(x\%K_p\right)}$ is periodic with period K_p. Thus, $\forall t, \int_t^{t+K_p} g'(u)\,du = c$, where c is a constant. By taking the integral from both sides of $g'\left(x+K_p\right) - g'(x) = 0$, while defining $h(x) = \int_{x_0}^{x} g'\,dx + h(x_0)$, we have $h\left(x+K_p\right) - h(x) = c$. If c is zero, then $h(x)$ is periodic. If $c \neq 0$, we define $v(x) = g'(x) - \frac{c}{K_p}$ and, without loss of

generality, we define $h(x)$ to be $\int_{x_0}^{x} v dx + h(x_0)$. Now $h\left(x+K_p\right) - h(x) = 0$, that is, $h(x)$ is periodic with period K_p. We have

$$\int_{x}^{x+K_p} v(x)\,dx = \int_{x}^{x+K_p} g'(x) + \frac{c}{K_p} dx = r(x) + \frac{c}{K_p} x$$

where $r(x)$ is a periodic function with period K_p. Replacing $g(x)$ with its original function $\frac{\hat{f} \restriction_{R_p}(x)}{I\left(x\%K_p\right)}$, we have

$$\frac{\hat{f} \restriction_{R_p}(x)}{I\left(x\%K_p\right)} = \frac{c}{K_p} x + r(x)$$

Thus, for all $0 \leq x_0 < K_p$ and for any natural number $n \in \mathbb{N}$ we have $I((x_0 + nK_p)\%K_p) = c_0$ where c_0 is a constant. Thus, $\forall 0 \leq x_0 < K_p, n \in \mathbb{N}$

$$\hat{f} \restriction_{R_p}(x_0 + nK_p) = \frac{c}{K_p}(x_0 + nK_p) * I((x_0 + nK_p)\%K_p) +$$

$$r(x_0 + nK_p) * I((x_0 + nK_p)\%K_p) =$$

$$C_0 + \frac{c}{K_p}(nK_p) * c_0 + r(x_0) * c_0 = b_0 + an$$

where $C_0 = \frac{c}{K_p} x_0 c_0$ and $b_0 = C_0 + r x_0$ are both constants. □

Next we show that there exists a set of local maxima from $\hat{f} \restriction_{R_p}$ that are linear and periodic.

Proposition 3. *Given time series T and an IoR $R_p \subset T$, then $\exists P = \{p_1, p_2, p_3, ...p_m\} \subset \hat{f} \restriction_{R_p}$, such that $p_1 < p_2 ... < p_m$ are periodic points and $p_i, 1 \leq i \leq m$ is a local maximum.*

Proof. Consider R_p to be an IoR with length p and period K_p, i.e., $\forall x \in R_p$, $f \restriction_{R_p}\left(x+K_p\right) = f \restriction_{R_p}(x)$. From the definition of autocorrelation we know that the autocorrelation of f, $\hat{f} \restriction_{R_p}(x) = \int_{R_p} f \restriction_{R_p}(\tau) f \restriction R_p(x+\tau)\,d\tau$, reaches its local maxima whenever the phase from $f(\tau)$ matches the phase from $f(x+\tau)$, or in other words, the signals of the recurrences in $f \restriction_{R_p}(\tau)$ and $f \restriction_{R_p}(x+\tau)$ exactly match each other. If we consider t_0 to be the first time $f \restriction_{R_p}(\tau)$ and $f \restriction_{R_p}(x+\tau)$ matched with each other, then $\forall i$ where

$iK_p \in R_p | \hat{f} \restriction_{R_p} (x)$ is a local maximum at $\hat{f} \restriction_{R_p} (iK_p)$. Then P from Proposition 4 is $P = \{\hat{f} \restriction_{R_p} (K_p), \hat{f} \restriction_{R_p} (2K_p), ... \hat{f} \restriction_{R_p} (mK_p)\}$. □

We can thus prove the properties that we use to differentiate an IoR from a non-IoR, namely, the linearity and periodicity of local maxima of an IoR.

Proposition 4. *Given time series T and an IoR $R_p \subset T$, then $\exists P = \{p_1, p_2, p_3, ... p_m\} \subset \hat{f} \restriction_{R_p}$, such that $p_1 < p_2 ... < p_m$ are linear and periodic points where $p_i, 1 \leq i \leq m$ is a local maximum.*

Proof. Proposition 3 shows how to find a set of local maxima from the autcorrelation of an IoR that is periodic. Proposition 4 follows Theorem 1 that proves the linearity of periodic points from the autocorrelation of an IoR. □

4.11.2 Noise analysis

Jitter

Jitter results from hardware clock limitations in reproducing the true periodic signal. In an ideal scenario, we would be able to sample the signal in exact τ intervals. However, jitter perturbates the sampling points. If the sampling occurs at τ intervals with sampling error ϵ then we have:

$$\hat{f}(n) = \sum_{m=0}^{N-1} f(m + \epsilon_m) f(n - m + \epsilon_{n-m}) \tag{4.5}$$

Given that ϵ is small, f can be estimated by its first order Taylor series expansion [147], i.e., $f(n+\epsilon) = f(n) + \epsilon f'(n)$ then:

$$\hat{f}(n) = \sum_{m=0}^{N-1} \big(f(m) + \epsilon_m f'(m)\big)\big(f(n-m) + \epsilon_{n-m} f'(n-m)\big)$$

Since ϵ_m is a random variable we have:

$$\mathbb{E}\big(\hat{f}(n)\big) = \mathbb{E}\Big(\sum_{m=0}^{N-1} f(m) f(n-m)\Big) + \sum_{m=0}^{N-1} \mathbb{E}(\epsilon_m) f'(m) f(n-m) + \sum_{m=0}^{N-1} \mathbb{E}\big(\epsilon_{n-m}\big) f'(n-m) f(m) + \sum_{m=0}^{N-1} \mathbb{E}\big(\epsilon_{n-m}\epsilon_m\big) f'(n-m) f'(m) \tag{4.6}$$

$$\mathrm{Var}\big(\hat{f}(n)\big) = \mathrm{Var}\Big(\sum_{m=0}^{N-1} \big(f(m) + \epsilon_m f'(m)\big)\big(f(n-m) + \epsilon_{n-m} f'(n-m)\big)\Big) \tag{4.7}$$

where $\mathbb{E}$ is the expectation and Var is the variance function. To calculate Equations 4.6 and 4.7 we need to know the distribution of ϵ_m functions and their covariance with each other, which are implementation dependent factors. We empirically demonstrate that our implementation is not highly sensitive to these two values (Section 4.9). If we assume ϵ_m are independent Gaussian variables with $\mathbb{E}\big(\epsilon_m\big) = 0$ and $\mathrm{Var}\big(\epsilon_m\big) = \eta$ we can simplify Equation 4.6 as follows [147]:

$$\mathbb{E}\big(\hat{f}(n)\big) = \mathbb{E}\Big(\sum_{m=0}^{N-1} f(m) f(n-m)\Big) \tag{4.8}$$

Noting that $\mathrm{Var}\big(f(n-m) f(m)\big) = 0$ and all ϵ_m are independent, then expanding Equation 4.7 results in a bounded variance:

$$\mathrm{Var}\big(\hat{f}(n)\big) \leq \eta \sum_{m=0}^{N-1} f'(m)^2 \sum_{m=0}^{N-1} f(n-m)^2 + \eta \sum_{m=0}^{N-1} f'(n-m)^2 \sum_{m=0}^{N-1} f(m)^2 + \eta^2 \sum_{m=0}^{N-1} f'(n-m)^2 \sum_{m=0}^{N-1} f'(m)^2 \tag{4.9}$$

The Gaussian sampling error model (Equation 4.8) shows that the expected line computed from the sampled signal is the same as the line from Proposition 4 with a variance that can be computed by Equation 4.9. In signal processing $\sum_{m=0}^{N-1} f(m)^2$ is defined as the energy of f over its domain. Since the domain of f in Equation 4.9 is restricted to the IoR, its energy is bounded. Thus, $\mathrm{Var}(\hat{f})$ is bounded by three values, the

noise variance η, the energy of f and its derivative f' during the IoR.

Amplitude noise

Amplitude noise is the small error present around each read of the sampled signal. This noise often results from two sources: (1) read errors, and (2) performance errors. The read error is an inseparable part of any digitised device where each read can be affected by environmental factors such as temperature or humidity. The second source of amplitude error comes from the signal's source where repeats cannot be performed 100% identically. This variation can affect our calculations. Amplitude noise is modelled as an error around each read, i.e.:

$$\hat{f}(n) = \sum_{m=0}^{N-1} (f(m) + \varepsilon_m)(f(n-m) + \varepsilon_{n-m}) \tag{4.10}$$

Using the same analysis from Section 4.11.2 with Gaussian zero mean noise and variance ϑ it follows that:

- $\mathbb{E}\big(\hat{f}(n)\big) = \sum_{m=0}^{N-1} f(m) f(n-m)$, and
- $\mathrm{Var}\big(\hat{f}(n)\big) \leq \vartheta F + \vartheta F'$

where F and F' are the area under the IoR and are thus bounded. Hence the computed line derived from measurements affected by Gaussian noise is expected to be the same as that in Proposition 4 with an error bounded by the noise variance and the area under the IoR.

Discretization error

Discretization error shows itself as variations in repetitions of the digitised SoR in an IoR. A sampling rate that is not synchronised to the period of repeat in an IoR introduces discretization error. In this case, since the period of repeat in the IoR is not factorable to

the sampling rate, samples are taken at different points from the successive repeats of the waveform. Thus, each digitised SoR from the IoR varies from one repeat to another even though all the original SoRs are identical in the IoR. If Δ is the sampling period and K_p is the repeat period from an IoR, we can define discretization error to be a random variable, γ, appearing in the period of repeat with respect to the sampling period as follows:

$$\exists l \mid K_p = l\Delta + \gamma \tag{4.11}$$

where l is an integer corresponding to the number of samples per repeat. Assuming we have R SoR in the IoR, we can rewrite the autocorrelation ($\hat{f}$) of an IoR to be:

$$\begin{aligned}\hat{f}(n) &= \sum_{m=0}^{Rl-1} f(m)f(n-m) = \sum_{j=0}^{R-1}\sum_{i=0}^{l-1} f(i+jK_p)f(n-i-jK_p) \\ &= \sum_{j=0}^{R-1}\sum_{i=0}^{l-1} f(i+jl\Delta+j\gamma_{i+jl\Delta})f(n-i-jl\Delta+j\gamma_{n-i-jl\Delta})\end{aligned}$$

By introducing a new variable $z := i + jl\Delta$ and a random variable $\zeta := j\gamma$ then we have

$$\hat{f}(n) = \sum_{z=0}^{N-1} f(z+\zeta_z)f(n-z+\zeta_{n-z}) \tag{4.12}$$

Comparing Equation 4.12 to Equation 4.5 we see that the two equations are identical. Thus we can perform the same analysis as we did in Section 4.11.2.

Variations from the period of repeat produced by the signal's source have the same effect as the unsynchronized sampling discretization error due to unsynchronized sampling. In these cases the signal's source is not able to generate every SoR in the IoR identically. For example, a trainee cannot perform each repeat of the same exercise with exactly the same speed. Thus, there exists an error around the periodicity of the IoR associated with that exercise.

In Equation 4.11, we assume that every SoR in the IoR has the same sampling length l.

This assumption imposes an upper bound on the sampling frequency. To calculate this upper bound, the number of samples from the shortest and the longest SoR in the IoR have to be equal, i.e., length$(S_{min,I})/\phi$ = length$(S_{max,I})/\phi$, where $S_{min,I}$ and $S_{max,I}$ are the shortest and longest SoR, respectively, from the IoR I, and ϕ is the sampling frequency.

4.11.3 Impact of noise on OToR

In this section we analyse the impact of white noise on OToR's performance in detecting and tracking consecutive repeats. For our experiment, we generate a synthetic time series of ten consecutive repeats of the SoR signal shown in red in Figure 4-13.

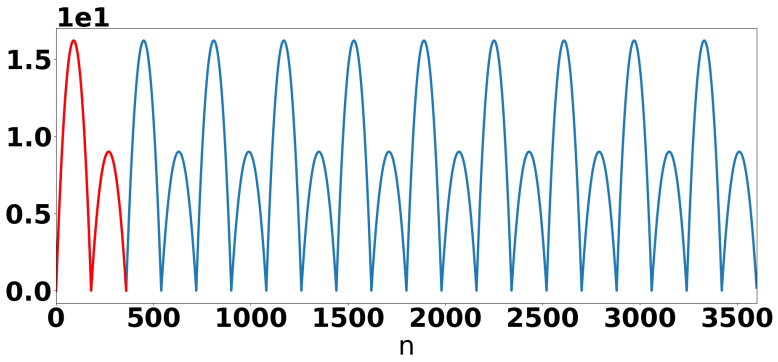


Figure 4-13: A synthetic IoR with ten repetitions of a SoR. The SoR (the red signal) is generated from two signals with the same period but different amplitude.

We added different levels of zero-mean white noise to the generated time series and ran the OToR algorithm over each generated time series. Figure 4-14 reports the maximum number of consecutive repeats detected by OToR at each noise level. In this figure the noise level is represented by signal-to-noise-ratio (SNR) on the X axis. The figure shows that noise level of up to 3:1 SNR has minimal effect on OToR's detection and tracking of SoR, i.e., OToR can handle SNR of 3:1 with at most two miscounting of repetitions. The figure shows that OToR performance drops for higher signal-to-noise-ratios. However, it's performance flattens at SNR 1:1 where it can detect and track five consecutive repeats before breaking the IoR region.

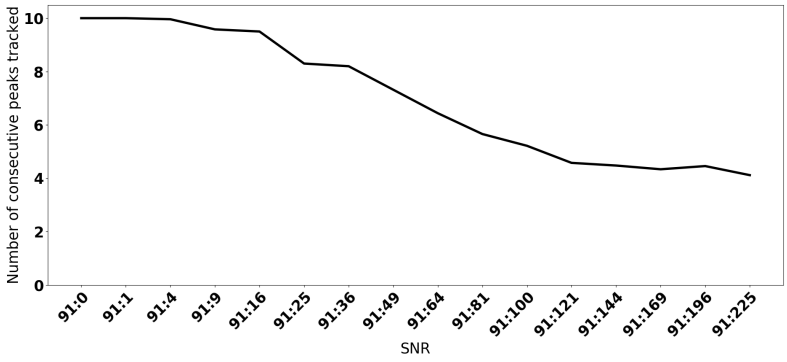


Figure 4-14: Maximum number of consecutive peaks detected and tracked by OToR in the presence of noise.

4.11.4 Detecting local maxima from autocorrelation

OToR is based on finding the local maxima from the autocorrelation of an incoming time series. In a noisy environment local maxima are happening more often due to the fluctuations resulted from adding noise to the signal. Thus, finding the local maxima of the autocorrelation in a noisy environment requires defining a neighbourhood in which the local maximum value is representing the actual local maxima, i.e., we need to find a window of the neighbourhood in which local maxima from the autocorrelation plus their noise value are masking other local maxima generated by the signal's fluctuations due to noise. We have demonstrated this effect using the experiments from Section I.C, Impact of noise on OToR. We ran OToR by using different neighbourhood sizes (N) for computing the local maxima. Figure 4-15 shows the number of consecutive repeats that OToR detects before breaking the IoR. The figure shows that selecting a small neighbourhood size ($N \leq 10$ points) results in a sharp drop in OToR's performance. However, setting the neighbourhood's size large enough ($N \geq 15$) results in a uniform performance from OToR, where its performance smoothly decreases until it flattens at noise levels of SNR= $1:1$ or higher.

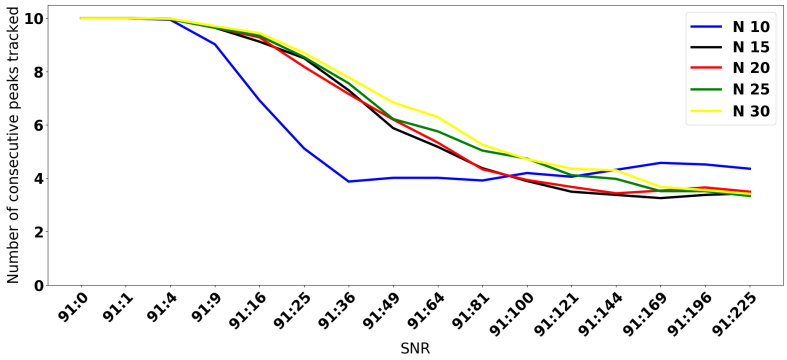


Figure 4-15: The effect of neighbourhood size (N) in detecting local maxima in the presence of noise.

4.11.5 Visualising OToR algorithm's behaviour for finding new IoR

In this section, we demonstrate how OToR detects an IoR in scenarios where a non-convex SoR is repeating inside an IoR. Consider a SoR that includes two identical signals, see Figure 4-16. In this example, SoR (shown in red) is generated from two identical signals, which are slightly apart from one another inside the SoR. Note that if the two identical signals ware not apart then by the definition SoR was defined as one of the signals. Thus, SoR is generated from the two identical signals with some distance in between. SoR is repeated three times in the time series. To find the Interval of Recurrence (IoR) generated by this SoR, OToR calculates the autocorrelation of the time series, shown on the right in Figure 4-16. OToR selects the last peak (marked as P_s in the figure) as the starting point and searches for an ordered set of points which are starting from P_s and are both periodic and aligned. OToR starts by searching the previous peaks in a heuristic approach. It first elects P_{11} and generates the line l_1 (shown in dashed). At this stage, the expected period is Θ_1. However, in the set of local maxima calculated from the autocorrelation, no point satisfies both periodicity (next point should be Θ_1 apart from P_{11}) and linearity (lies on l_1) conditions. Next, OToR skips P_{11} and elects P_{21} to generate the line l_2. In this iteration, the expected period is Θ_2. Thus, OToR checks the points that are Θ_2 apart from P_{21} and elects P_{22}. OToR confirms the linearity condition of the three points (P_s, P_{21}, and P_{22}) by

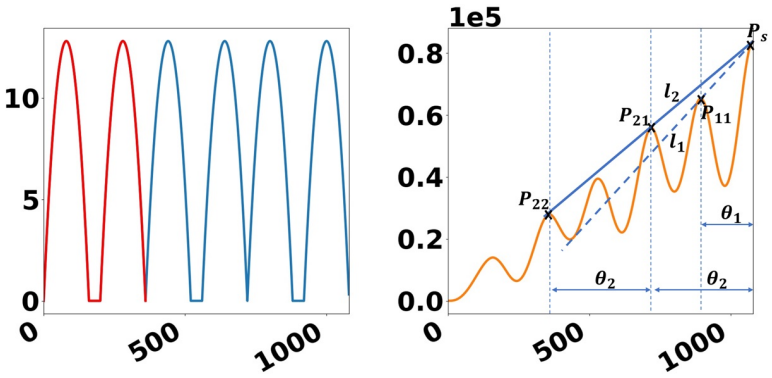


Figure 4-16: Signal of recurrence with identical signals. On the left an IoR with three repetitions of the SoR (shown in red) is plotted. On the right the autocorrelation of the IoR is shown.

checking that p_{22} is placed on the pre-calculated line, l_2.

Thus, OToR selects P_{22}, P_{21}, and P_s as the three peaks defining the IoR in the time series. From now on, OToR iteratively tracks incoming local maxima with $n * \Theta_2$ period apart from P_s that are aligned on l_2 starting from $n = 1$. This step is shown in Figure 4-17 where l_2 extension is dashed. P_n is the new local maximum calculated from the autocorrelation that is Θ_2 distance apart from P_s.

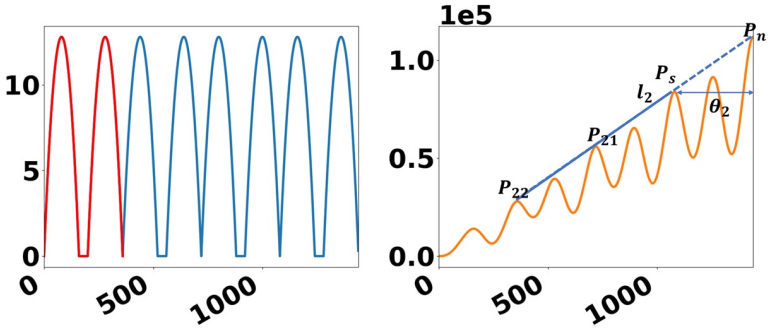


Figure 4-17: OToR tracks SoR in the detected IoR from Figure 4-16. On the left the same IoR with four repetitions of the SoR (shown in red) is plotted. On the right the autocorrelation of the IoR is shown. OToR tracks the IoR by l_2 extension which is shown in dashed.

Our next example concerns a SoR which is generated from two periodic signals with an identical period but different amplitudes, Figure 4-18. As shown in the image, OToR's

behaviour is similar to its response in the previous example, i.e., OToR dismisses the first line (l_1) and then tracks the points lined on l_2.

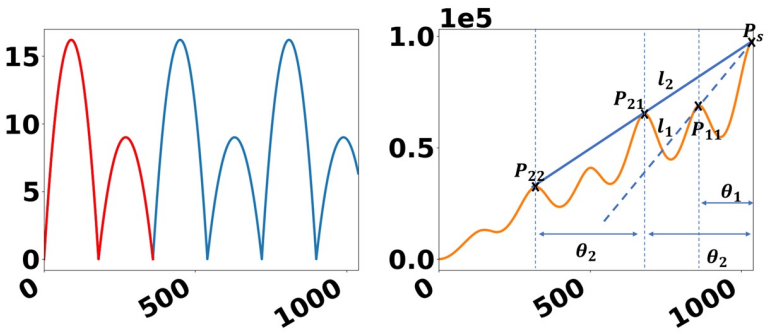


Figure 4-18: Signal of recurrence generated from two signals with same period but different amplitude. On the left an IoR with three repetitions of the SoR (shown in red) is plotted. On the right the autocorrelation of the IoR is shown.

Chapter 5

Analysing waveforms by their shape

5.1 Introduction

In the previous chapter, we discussed how to extract exercise segments from an incoming time series of measurements from IMU sensors. As discussed in Section 3.2, the next two steps to analyse weight training exercises are to (1) Compute the average time series, known as a prototype, from the observed time series, and (2) Compare time series to the computed prototype.

Our weight training application motivated the idea to analyse time series through their visual "*shape*", i.e., using a method to analyse time series through their appearance. Comparing time series by their appearance or visual *shape* has diverse applications, such as classifying heartbeat ECG records into normal and abnormal signals [148, 149, 150, 151], clustering accelerometer records from wearables attached to a trainee into different sets of exercises [152], or categorising data received from a space shuttle [153]. In these problems, the relationships between points on the time series are more important than each individual absolute value. For example, Figure 5-1 shows an abnormal and a normal heartbeat from the ECG200 database [2]. We can see that both heartbeats have peaks in the highlighted areas. However, the peaks in the normal heartbeat are much smoother

than the peaks in the abnormal one. This smoothness manifests itself in the relationship between the peak and its neighbourhood values.

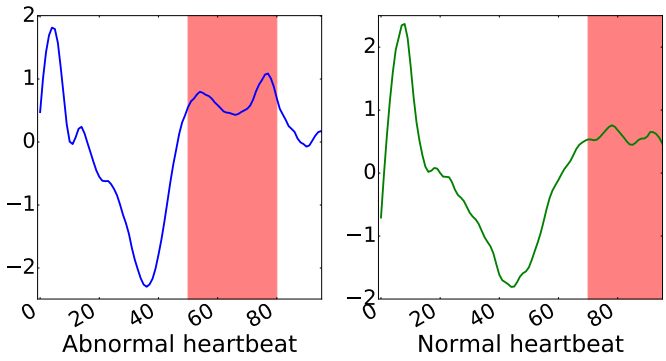
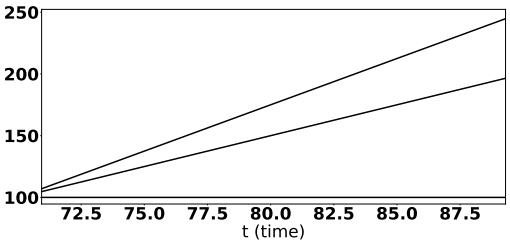


Figure 5-1: Abnormal and normal waveforms of heartbeats from ECG200 data set [2].

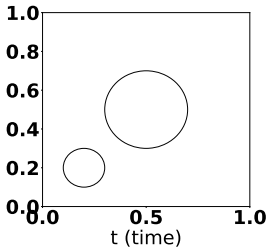
Traditionally, shape analysis for time series data has focused on comparisons between pairs of time series. In such an approach, shape is indirectly characterised by a mapping between smaller pieces of the two time series to each other. For example, dynamic time warping (DTW), the most common distance measure between pairs of time series for shape analysis, uses dynamic programming to find the "best" mapping between the two time series [154, 155]. A DTW value is a numeric distance measure that evaluates the dissimilarity between two time series. An alternative approach identifies primitives from a time series where a shape is considered to be decomposable into peaks and troughs. In this approach, similar shapes are assumed to behave similarly through time, i.e., if one goes up the other goes up and if one goes down the other goes down as well. In this case, the similarity between two time series is defined through correlation [139, 61]. Although these approaches can be used to define the similarity of time series from a set of data points, they do not provide any insights about the internal relationships among the points of a single time series. This insight is crucial for the interpretation of time series data. As shown in Figure 5-1, the notion of the shape of a time series would reveal the underlying relationship among the points on the time series, such as the smoothness of peaks shown in the figure. Our approach is to define the notion of shape in such a way

that it enables us to reconstruct the original time series exactly. In this way, a time series is modelled as a unique geometric curve that can be presented to an observer like any other geometric object such as a line, circle or cycloid.

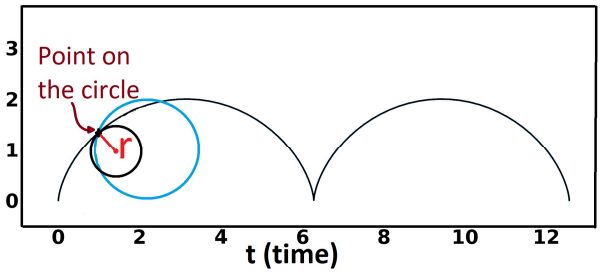
Mathematically, a time series can be described as a curve in space. The simplest curve is a line, an object with no curvature (peaks and troughs) (Figure 5-2a). Another simple object is one with constant curvature—a circle (Figure 5-2b). Third, consider a cycloid; the curve generated by a point on the circumference of a circle that rolls along a straight line, (Figure 5-2c). The cycloid's curvature changes according to the fixed point on the circle. Differential geometry generalises this idea to uniquely determine the shape of the curve through its curvature [156]. The curvature shows the degree of deviation while



(a) Lines are objects with zero curvature.



(b) Circles are objects with constant curvature.



(c) A cycloid is the curve traced by a point on the rim of a circular wheel (blue circle) as the wheel rolls along a straight line.

Figure 5-2: Shapes are described by the amount of curvature (defined by the radius (r) of the tangent circle) they have at any point.

moving along the curve from a flat line. By knowing the curvature at each point we can always generate the same curve regardless of the initial position [157]. This definition of shape brings three important characteristics for shape analysis, namely:

1. Scale invariance,
2. Translation invariance, and
3. Rotation invariance.

Our aim is to use differential geometric properties of curves to introduce a new representation of a time series that uniquely identifies the time series by its shape. The Fourier analysis of this representation leads us to design a vector space where each time series is uniquely defined by a vector. The goal is to address two fundamental problems in time series analysis: (1) How to compute a prototype from a given set of time series?, and (2) How to compare a pair of time series? We use two important tasks in time series analysis: (1) *K-Means clustering* and (2) *Nearest Centroid Classifier* (NCC) to demonstrate the effectiveness of these methods in addressing these two fundamental problems.

Comparing time series is an important problem when trying to group similar time series. To determine whether a given time series is from a particular group, a common approach is to compare the new time series to every member of the group, which is an inefficient approach. This is the main reason behind the second problem that we consider in time series analysis where a prototype is needed to represent a group of time series. A centroid for a group of time series can be seen as the mean value for the given group, i.e., the centroid is a data point (time series) that has the minimum average distance to every other member of the group. Computing a centroid is coupled with designing a distance measure. The mean property of a centroid enables various important types of analysis on time series, such as: (1) Detecting anomalies in a semi-supervised fashion [158, 159]. (2) Designing a fast indexing mechanism for data base retrieval [160, 161]. (3) Generating synthetic time series data [162, 163]. (4) Designing loss functions for optimisation problems with applications in deep neural networks [164, 165].

In summary, our contributions in this chapter are:

- **Novel method to represent the shape of a time series:**We Design a new represen-

tation for time series based on differential geometry to capture the shape of a time series.

- **Novel vector space:** We develop a vector space that can effectively represent the shape of a time series, and prove several key properties of this space.
- **Shape prototype:** We provide a theoretical framework to efficiently compute the shape prototype for a set of time series using the shape representation.
- **Effectiveness:** We empirically show using 48 data sets that the average shape (centroid) designed in this chapter together with our distance metric is the most effective method in comparison to three state-of-the-art methods when classifying time series data using a NCC for analysing time series by their shape. Equally important, the quality of the clusters found by our approach compares favourably with clusters found by the best-known technique.
- **Efficiency:** We show that the time complexity of our method is lower than the comparison methods, and empirically confirm that Contour (the method designed in this chapter) is the most efficient method for computing the shape prototype to be used in both clustering and classification of time series data when shape is the differentiating feature.

5.2 Problem statement

In this chapter the aim is to define a time series as a discrete geometric object generated by sampling a continuous planar curve. Thus, only finite, planar time series are considered. To define the geometric shape of a time series, we review the mathematical definition of a time series and then review the definition of shape through curvature from geometry.

Time series: A time series T_s of length n is a finite ordered set of measurements, say $\{Y(t_k)\}$ through time starting from time t_1 and ending at time t_n: $T_s = \{Y(t_1), Y(t_2), ..., Y(t_n)\}$.

To identify the shape of a time series, we represent it using the concept of curvature, i.e., the degree to which a curve deviates from a line. The curvature defines the velocity of the rotation of the tangent to the curve while moving along the curve.

The curvature at a point is defined by the relationship between the tangent vector to the curve and its second derivative vector at that point [166]. Formally, the curvature of a twice differentiable planar curve $Y(X)$ at a point $X^\star$ is:

Curvature κ:

$$\kappa_Y(X^\star) = \frac{Y''(X^\star)}{\left(1 + Y'(X^\star)^2\right)^{3/2}} \tag{5.1}$$

Anderson and Bezdek showed that it is possible to estimate the curvature from a discrete curve [156]. Since then, there have been many studies to estimate the curvature for discrete curves [167, 168, 169]. We use the simplest definition of curvature to perform our experiments, i.e., $Y' = \Delta Y / \Delta X^\star$ and $Y'' = \Delta Y' / \Delta X^\star$. Note that our method is independent from any particular form of approximation, and thus, can be used with any of the curvature estimators.

Curvature measures the amount of deviation from a flat line at each point along the curve. We use this notion to define the shape of a time series as follows.

Definition 1. Shape of a time series: The shape $S_{T_s}(t_i)$ of a time series T_s at point t_i is the total curvature of measurements in T_s from t_1 to t_i.

$$S_{T_s}(t_i) = \int_{t_1}^{t_i} k_Y(t)\,\mathrm{d}t \tag{5.2}$$

The definition of shape in Equation 5.2 is a way to linearise $T_s = \{Y(t_1), \ldots Y(t_n)\}$. Consider the interval $[t_1, t_n]$ and define $\Delta t_i = t_i - t_1$, where $t_i \in [t_1, t_n]$. If $\Delta t_i \to 0$ then $S_{T_s}(t_i)$ is estimated by the tangent vector at point t_i, i.e., the direction that the curve Y should move from point t_1 to point t_i in the interval $[t_1, t_n]$. From a global point of view, $S_{T_s}(t_i)$ reflects the total amount of turning Y takes to reach point t_i from point t_1.

Since any window of a time series is also a time series, this definition of shape can be

applied recursively to any segment of a time series. This property leads us to define the **Shape-Series** for any given time series T_s.

Definition 2. Shape-Series S_{T_s}: A Shape-Series S_{T_s} of a time series T_s of length n, is an ordered set of shapes with n members such that each value in the series S_{T_s} at t_i is associated with one and only one value in the time series and represents the shape of the time series at t_i, i.e., $S_{T_s} = \{S_{T_s}(t_1), S_{T_s}(t_2), ..., S_{T_s}(t_n)\}$

An example of calculating a Shape-Series is shown in Figure 5-3. In this figure we show the two steps to compute the Shape-Series of $\sin x$. It starts by computing the curvature of the curve at each t_i in the first step. The relation between curvature and the curve can be seen as a map for tracing the original curve in a 2-dimensional space. We start by moving forward while rotating. The amount of rotation at each point is given by the curvature at that point. In the second step, we perform a cumulative sum over the curvature to compute the Shape-Series. The Shape-Series tells us how much rotation in total we have performed at each point.

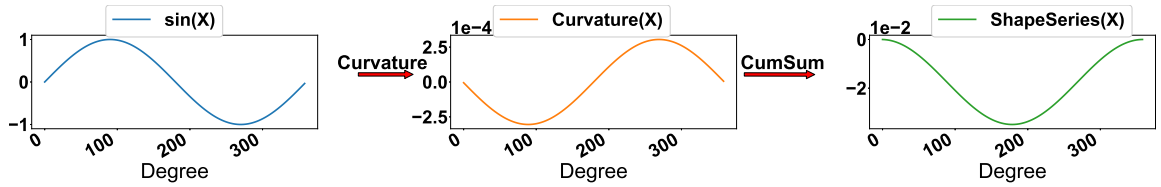


Figure 5-3: Steps to compute a Shape-Series for a given time series. First, we compute the curvature of each point. Second, we compute Equation 5.2 for each point.

5.3 Shape similarity

The Shape-Series (Definition 2) quantifies the shape of a curve and can be used to compare two curves. We first show in Theorem 5 that each shape quantity uniquely identifies its associated curve modulo translations and rotations.

Proposition 5 (Shape uniqueness proposition)**.** *Given a Shape-Series of a planar curve on the interval* $[t_1, t_n]$*, there is a unique translation- and rotation-invariant curve that is quantified by the Shape-Series.*

Proof. From the *first fundamental theorem of calculus* and the definition of the shape function, i.e., $S_{T_s}(t) = \int_{t_1}^{t} k_Y(t)\,\mathrm{d}t$, we have $k_Y(t) = \dfrac{dS_{T_s}(t)}{dt}$. Proposition 5 is a direct consequence of the *fundamental existence and uniqueness theorem for space curve* (Theorem 5.3, Lipschutz [157]), which proves that there is exactly one planar curve associated with any given curvature function on the plane, modulo any translation or rotation. Thus, for any arbitrary Shape-Series there is exactly one curve on the plane for which the derivative of the Shape-Series is the curvature of that curve. □

Proposition 5 guarantees the uniqueness of the shape associated with a shape function. We use this uniqueness property to define a new vector system to represent shapes. We use the completeness characteristics of the Fourier transform—i.e., it is an invertible linear transformation—to design the shape vector system.

The DFTs of any two Shape-Series of length n yield a pair of (frequency) vectors in R^n. Specifically,

$$\vec{V}_{S_{T_s}} = \vec{\mathrm{DFT}} * \vec{S_{T_s}} \tag{5.3}$$

where $\vec{DFT}$ is the Discrete Fourier Transform matrix, $*$ defines the outer product of two vectors and $\vec{S_{T_s}}$ is the Shape-Series of the time series in a vector form. We define the distance between these two vectors, which we call the *Angular Shape Distance* (ASD), to be the angle between the vectorization of their Shape-Series.

$$\mathrm{ASD}(T_s, T_q) = \cos^{-1}\left(\frac{\vec{V}_{S_{T_s}} \cdot \vec{V}_{S_{T_q}}}{\|\vec{V}_{S_{T_s}}\| \|\vec{V}_{S_{T_q}}\|}\right) \tag{5.4}$$

where $\cdot$ is the Euclidean dot product (Algorithm 5). Note that Equation 5.4 is an angular measure that ranges from 0 to 2π. Algorithm 5 shows how to compute the angular

similarity between two Shape-Series.

Algorithm 5: Find Angular Shape Distance between two time series

1 **Function** ASD(A, B)**:**
  Data: A: time series of length n, B:time series of length n
  Result: Return distance value between [0, 180]
2 NA ← A - MEAN(A);
3 NB ← B - MEAN(B);
4 ShapeA ← SHAPESERIES(NA);
5 ShapeB ← SHAPESERIES(NB);
6 fA ← DFFT(ShapeA); // DFFT returns discrete Fast Fourier Transform
7 fB ← DFFT(ShapeB);
8 $D \leftarrow \cos^{-1}\left(\frac{\mathbb{R}(fA.\overline{fB})}{\|fA\|\|fB)\|}\right)$; // $\mathbb{R}$ shows the real part of DFT and $\overline{fB}$ is the conjugate of fB. $\|\|$ is the norm operator
9 **return** D;
1 **Function** SHAPESERIES(T)**:**
  Data: T: time series of length $T = \{Y(t_1), ..., Y(t_n)\}$
  Result: Return Shape-Series of the given time series
2 CurveT ← []; // List of curvature for each point in T
3 **for** $i \leftarrow$ **to** n **do**
4 CurveT[t_i] ← K($Y(t_i)$); // K returns curvature of a given point.
5 **end**
6 Res ← CUMSUM((CurveT));
7 **return** Res;

Algorithm 5 starts by making each waveform zero-mean (lines 2 and 3). It calculates the Shape-Series for each normalised waveform (lines 4 and 5). Lines 6 to 8 are where the ASD is computed for the given waveforms. The SHAPESERIES function computes the Shape-Series for a given waveform by first calculating the curvature for each point on the waveform (lines 3 and 4). It then calculates the cumulative sum over the curvature's result (line 6).

To comply with geometric shape properties, a true shape comparison should be (1) *scale invariant* and (2) *translation invariant*. The scale invariance property ensures that every shape is identical regardless of magnitude, e.g., a circle is always a circle regardless of its radius. The translation invariance or time shift invariance makes sure that curves are identical and independent of both their occurrence in time and their observation

time. Two shapes are considered similar if one can be obtained by uniformly scaling or mirroring the other one (see Chapter 5 in Pedoe [170]).

The scale invariant property of ASD comes from the linearity property of the curvature and the cumulative sum, which make Shape-Series linear to scaling. The linearity property of the discrete Fourier transformation, i.e., $DFT\left(aS_{T_s}\right) = aDFT\left(S_{T_s}\right)$ and the fact that angles between vectors are invariant to the length of the vectors makes ASD scale invariant. The shift and rotation invariant properties of ASD result from curvature's definition that is invariant to position and rotation. Therefore, ASD is translation and rotation invariant.

Definition 3. Shape-Sphere is the identity sphere generated by normalising all the vectors in the Shape Space. Because the similarity between vectors in the Shape vector is defined as the angle between them, without loss of generality, we can normalise any Shape-Series vector $\vec{V_{S_{T_s}}}$ and transform it to Shape-Sphere:

$$\vec{NV}_{S_{T_s}} = \frac{\vec{V_{S_{T_s}}}}{\|\vec{V_{S_{T_s}}}\|} \tag{5.5}$$

Figure 5-4 shows the result of this transformation.

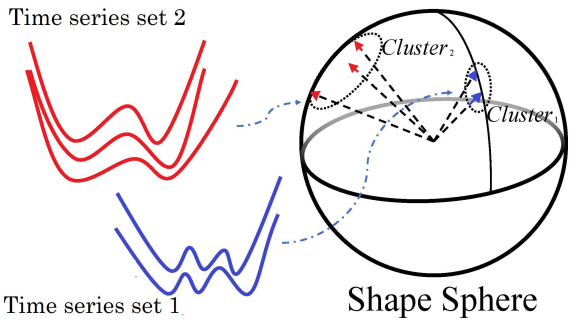


Figure 5-4: Shape-Sphere for time series analysis.

One of the advantages of ASD is that it induces a pseudometric on shape space. We prove the metric properties of ASD in Section 5.3.1.

5.3.1 The metric property of ASD

Metrics play an important role in clustering and classification algorithms, where the goal is to assign similar objects to the same group while assigning dissimilar objects to different groups. We prove that *ASD* is a pseudometric. We use this property to design an efficient and accurate method for analysing time series. The metric properties of ASD are as follows:

1. $ASD\left(T_s, T_q\right) \geq 0$.
2. $ASD\left(T_s, T_q\right) = ASD\left(T_q, T_s\right)$.
3. $ASD\left(T_s, T_q\right) = 0 \iff T_s = \Gamma\left(aT_q\right)$ for some constant factor $a > 0$ and some right shift associated with Γ, the translation function.
4. *ASD* is translation invariant, i.e., $ASD\left(T_s, T_q\right) = ASD\left(T_s, \Gamma\left(T_q\right)\right)$.
5. $ASD\left(T_s, T_q\right) \leq ASD\left(T_s, T_p\right) + ASD\left(T_p, T_q\right)$, i.e., the similarity measure *ASD* satisfies the triangle inequality.

Proof. The first and second properties result from the symmetric and positive semi-definite properties of angular similarity. The third and fourth properties follow from the definition of similarity.

The fifth property results from the uniqueness and one-to-one properties of the similarity function *ASD*, cf. Proposition 5. We showed that each shape function is uniquely identified by a vector in the frequency coordinates. Thus, for any arbitrary time series T_i there exists one and only one vector $(\vec{V_{S_{T_i}}})$ in the frequency coordinate that is associated with T_i. Without loss of generality, we transform $\vec{V_{S_{T_i}}}$ to Shape-Sphere $(\vec{NV}_{S_{T_i}})$. For any two vectors $\vec{NV}_{S_{T_s}}$ and $\vec{NV}_{S_{T_q}}$, which are not directly opposite one another on the shape sphere, we can define a unique circle centered at the origin that connects them. *ASD* is the value of the central angle of this circle. Any arbitrary third time series T_p can be mapped to the same frequency coordinate to get vector $\vec{NV}_{S_{T_p}}$. We then have three

possible positions for this new vector as follows:

1. $\vec{NV}_{S_{T_p}}$ is positioned on the arc associated with the central angle between $\vec{NV}_{S_{T_s}}$ and $\vec{NV}_{S_{T_q}}$. In this case, the angle between $\vec{NV}_{S_{T_s}}$ and $\vec{NV}_{S_{T_p}}$ plus the angle between $\vec{NV}_{S_{T_p}}$ and $\vec{NV}_{S_{T_q}}$ is equal to the angle between $\vec{NV}_{S_{T_s}}$ and $\vec{NV}_{S_{T_q}}$. Thus $ASD\left(T_s, T_q\right) = ASD\left(T_s, T_p\right) + ASD\left(T_p, T_q\right)$.
2. $\vec{NV}_{S_{T_p}}$ is positioned outside the arc associated with the central angle between $\vec{NV}_{S_{T_s}}$ and $\vec{NV}_{S_{T_q}}$ but on the same circle. In this case, the angle between $\vec{NV}_{S_{T_s}}$ and $\vec{NV}_{S_{T_p}}$ plus the angle between $\vec{NV}_{S_{T_p}}$ and $\vec{NV}_{S_{T_q}}$ is greater than or equal to the angle between $\vec{NV}_{S_{T_s}}$ and $\vec{NV}_{S_{T_q}}$. The same case happens when the angle between $\vec{NV}_{S_{T_s}}$ and $\vec{NV}_{S_{T_q}}$ is equal to 180°. Thus $ASD\left(T_s, T_q\right) \leq ASD\left(T_s, T_p\right) + ASD\left(T_p, T_q\right)$.
3. $\vec{NV}_{S_{T_p}}$ is positioned outside the arc associated with the central angle between $\vec{NV}_{S_{T_s}}$ and $\vec{NV}_{S_{T_q}}$ and on a different circle. Using the triangle inequality on the sphere created from the three circles joining each pair of vectors, we get that $\widehat{NV_1NV_3} + \widehat{NV_3NV_2} > \widehat{NV_1NV_2} = ASD\left(T_s, T_q\right)$. Therefore, $ASD\left(T_s, T_q\right) < ASD\left(T_s, T_p\right) + ASD\left(T_p, T_q\right)$.

Properties 2, 3 and 5 establish ASD as a pseudometric. As a result Shape Sphere is a pseudometric space. □

5.4 Computing the average shape—Contour prototypes

We define the average shape of a set of time series as the set's **contour**, as follows:

Definition 4. Contour: We define the contour of a set of time series as the Shape-Series that has the maximum similarity (minimum distance) to every Shape-Series member of a set of time series. Thus, the contour, C, of a set of time series, $T = \{T_{iS} | \forall i, 1 \leq i \leq n\}$ is the vector calculated by averaging all the vectors from the set of vectors V_s that represents T_s

in its corresponding Shape-Sphere:

$$C\Big(\{V_s\}\Big) = \frac{\sum_{i=1}^{n} \vec{V_{S_{T_{iS}}}}}{n} \tag{5.6}$$

To make contours free of any translation we use the periodic property of the Fourier transform (FT). A function with a DFT can be seen as a curve drawn on the surface of a cylinder, where the start and end points meet at the same location. Therefore, any time shift is equal to some rotation of the cylinder. To find the actual shift between two signals, we repeat one of the signals consecutively and take the signal-correlation between this new signal and the second signal. The amount of shift between the two signals is equal to the point where the computed correlation is maximised. Thus, we right-shift-rotate the second signal by the amount of shift resulting from the correlation to find the best match between the two signals. We perform this task as a preprocessing step for a given set of time series to calculate the average shape of a set. This process is shown in Algorithm 6. Procedure GETCENTROID first finds an average contour for a given set (line 2). It then right shifts all the time series in a given set to match the calculated centroid (line 3). The algorithm calculates the centroid from the shifted set as the Contour of the set (line 4). The GETCENTROID function computes the Fourier transform for every single member of a given set (lines 4 to 6) and return the average of all the Fourier transforms (line 7). The SHIFTBYCONTOUR right shift rotates a given set according to a given waveform. For each member of the set, it first finds the shift that matches the member with the given waveform. To perform this task, it finds the global maximum location from the cross correlation of the waveform and the member (lines 6 and 7) and right shift rotate each member by this amount (line 8). It returns the new waveforms at line 10.

Algorithm 6: Procedure Contour for a given set of Shape-Series

```
1  Function CONTOUR(S):
       Data: S: A set of Shape-Series of length n.
       Result: Returns the prototype for set S
2      cntr ← GETCENTROID(S);
3      shiftFreeS ← SHIFTBYCONTOUR(S, cntr);
4      C ← GETCENTROID(shiftFreeS);
5      return C;
1  Function GETCENTROID(S):
       /* Compute Average Centre for a given set of Shape-Series          */
       Data: S: A set of Shape-Series of length n.
       Result: Returns the contour for set S
2      m ← ‖S‖;
3      fftS ← [];
4      for i ← 1 to m do
5          fftS[i] ← DFFT(S(i));
6      end
7      cntr ← COLLAVERAGE(fftS);          // CollAverage returns the column average
                                             for the input matrix
8      return cntr;
1  Function SHIFTBYCONTOUR(S, C):
       /* Align Shape-Series according to a contour                       */
       Data: S: A set of Shape-Series of length n, C: A Shape-Series of length n
       Result: S′ a set of Shape-Series with every member shifted according to C
2      m ← ‖S‖;
3      med ←cntr.APPEND(cntr);
4      result ← [];
5      for i ← 1 to m do
6          AC ← CORELATION(S[i], m);
7          r ← ARGMAX(AC);
8          result.APPEND(rShfitRight(S[i], r);   // rShiftRight right rotate shift the
                                                    passed vector r times
9      end
10     return result ;
```

5.4.1 Complexity

Given a set of time series with m members of length n, computing the contour starts by calculating the shape representation of each member of the cluster. Because the shape representation is the result of a convolution, this takes $O(n \log n)$ time for each member. There are m members in the set, thus the whole process takes $O(mn \log n)$. The second step is to transform the shape representation into Shape-Sphere which is equivalent

to taking the Fourier transformation of each shape representation. However, since we have already calculated the shape representation through the Convolution Theorem, we already have this result. To compute the contour, we take the average from the set in Shape-Sphere, which takes $O(mn)$ time. After calculating the contour for the first time, we need to calculate the shift according to the correlation between the signals, which again, using the Convolution Theorem, takes $O\left(mn\log n\right)$. After adjusting all the data points in the cluster, we once again calculate the contour, which takes $O(mn)$. Thus, in total, calculating the prototype for a cluster takes $O\left(mn\log n+mn+mn\log n+mn\right)=O\left(mn\log n\right)$. Therefore, calculating the prototype for a cluster of m data points of length n is $O\left(mn\log n\right)$.

5.5 Comparison of centroids

In Proposition 5 we discussed the one-to-one relationship between a curve and its curvature translation. We transfer the curvature using a cumulative sum of the curvature series. The cumulative sum is equivalent to taking the integral of the curvature series. The result is a unique series up to a constant factor. This uniqueness allows us to analyse time series in the new feature space. This approach is different to the scheme used by DBA [124], cDBA [126] and SE in K-Shape [61] where a set of time series is represented by a prototypical time series in the input feature space. Instead, we compare time series to prototypes indirectly with the NCC. Specifically, we subdivide each subset of various labelled time series into training and test sets. Then we apply the four algorithms to the training data to secure prototypes for each subset. Finally, we compute the observed error rate (number of labelling errors) committed on the test data by each of the four NCC classifiers. The centroid that results in the highest accuracy is presumably the best representative of each labelled subset.

5.5.1 Baseline methods: DBA, cDBA and K-Shape

DBA is an averaging method based on Dynamic Time Warping (DTW). Gupta et al. showed that finding the global optimum prototype using DTW is an NP-Complete problem [123, 124]. This occurs because finding the optimum prototype is sensitive to the order of presentation of the inputs. Petitjean et al. introduced DBA, an iterative heuristic approach to overcome the ordering problem [124]. The initialisation of DBA chooses a random curve in a group. The objective function that monitors the progress of DBA and K-Means clustering is the sum of squared errors between the prototypes and the input data. Let $X = \{x_1, \ldots x_N\}$ be a set of unlabelled waveforms in feature space R^p(i.e., each x_j is a time series of p measurements); let $V = \{v_1, \ldots v_k\}$ in R^p be a set of k prototypes for the waveforms in X. Each V_j represents a subset X_j from X. Then the within group sum of squared errors (WGSS) between the input data and the prototypes, with respect to any vector norm $\|*\|$ on R^p is,

$$WGSS(V) = \sum_j \sum_i \|x_{ji} - v_j\|^2 \tag{5.7}$$

where x_{ji} is a member in X_i. In each iteration, DBA computes a prototype that lowers the within group sum of squared errors. In their most recent approach, Petitjean et al. introduced the idea of finding multiple prototypes for a given group. In this approach, the input space is divided into multiple sub-regions and then a prototype for each sub-region is computed [125]. This technique performs better for classification/clustering of data than the classical K-Means method. The sub-partitioning tries to produce convex hulls where an average is well defined (the average always appears inside the convex hull of its generators). All the techniques that we develop can also be used to calculate a prototype for any sub-partitioning of data and they all benefit from such sub partitioning. Thus, we only compare our method with DBA, the underlying method for computing a prototype for a given set of time series.

Constraint Dynamic time warping Barycenter Averaging (cDBA) Morel et al. introduced constraint dynamic time warping for averaging time series [126]. Their method is very close to DBA with an extra constraint on mapping through DTW. The authors showed that in cases where zero-mean normalisation cannot be performed over the data, DBA fails to create an accurate prototype. They proposed using locally constrained DBA (cDBA) defined by Muller [127] to improve the DBA prototype in non-normalised datasets. They empirically showed that the prototype computed using locally constrained DTW improves the DBA results in a classification task. Despite our best efforts, we were unable to obtain the original implementation of the cDBA software, so we implemented their method ourselves. It is important to note that although we obtained the same results as reported by the authors [126] on the five data sets used below, we were unable to replicate their speed results. Though the authors claim to have achieved the same speed as DBA, in our tests the algorithm was slower than DBA by a constant magnitude factor.

K-Shape Paparizzos et al. introduced K-Shape as a method for clustering time series using two notions about shape, namely: (1) Shape-Based Distance (SBD); and (2) Shape Extraction (SE). These are combined in [61] as Algorithm 3, which uses the same alternating estimation scheme as k-Means but with SBD and SE instead of Euclidean distance and vector averaging.

SBD is a pseudo-metric to compare a pair of time series [61]. SBD considers each time series as a signal, and defines the distance between two time series to be equal to one minus the correlation between the two time series. By normalising this value, the distance between two signals becomes a value in $[-1,1]$. To have an always positive distance the authors define the distance between two signals to be one minus the normalised maximum cross-correlation between the two signals [61]. Thus SBD ranges from 0 to 2, where 0 means perfect alignment.

K-Shape proposed SE for computing a prototype from a set of time series [61]. This scheme is based on the SBD distance, and attempts to find an optimal prototype for a

cluster using signal correlation maximisation. The main idea is based on the convolution operator, which creates an inner product space and thus can be used as a pseudo-distance. In this method, each z-normalised member of a given class is considered to be a vector in R^n where n is the number of features in the time series. In this approach, signal correlation is used as the distance between pairs of vectors. Let $X = \{x_1, \ldots x_N\}$ be a set of unlabelled waveforms in feature space R^n that generates a matrix of size $N * n$. Thus, for a given subset of vectors in R^n, the eigenvector associated with the biggest eigenvalue defines the dominant orientation of the spread of the vectors in R^n. Since every vector is z-normalised, the z-normalised eigenvector associated with the biggest eigenvalue of the given subset can be used as the prototype of the subset. Papparizos et al. showed how to compute this eigenvector using a few linear transformations [61]. The advantage of K-Shape's distance method over DTW is that it uses properties from the convolution theorem to design an efficient algorithm for finding prototypes. This gives SE a time complexity advantage over DBA when used in pattern recognition algorithms such as K-Means and Nearest Centroid Classifiers (NCC).

5.6 Experimental evaluation

In this section we compare DBA, cDBA and K-Shape to our approach using two partitions generated by two schemes: (1) Classification using the Nearest Centroid Classifier (NCC), and (2) Clustering using the K-Means clustering algorithm. We empirically evaluated our method in an experiment using 48 publicly available datasets [171]. Our goal is to answer two questions: (1) *How does the NCC classifier based on the contours extracted using the* CONTOUR *procedure compare to NCCs based on the prototypes computed using the other three state-of-the-art techniques for analysing time series data?* (2) *Is our approach more time efficient than the alternatives?*

We evaluate the four algorithms using the adjusted Rand index (ARI), a well-known scalar measure introduced by Hubert and Arabie [172] that compares pairs of partitions on n objects $X = \{x_1, \ldots, x_n\}$. Let $V(X)$ be a reference partition (ground truth partition) that purports to represent the "true structure" in X. Let $U(X)$ be a partition of X found with any clustering or classification algorithm named u. Then, $\text{ARI}\big(U(X), V(X)\big)$ measures the extent to which the labels in $U(X)$ match the ground truth labels in $V(X)$. The ARI maximizes at 1 when $U(X) = V(X)$, i.e., when all the labels in the two partitions agree. On the other hand, disagreement grows as ARI decreases, and at ARI= 0, there are no label matches.

The ARI can be used to compare the efficacy of different algorithms for clustering and classification. Specifically, let $U(X)$ and $W(X)$ denote partitions obtained by algorithms u and w. We first compute the pair $(x, y) = \Big(\text{ARI}\big(V(X), U(X)\big), \text{ARI}\big(V(X), W(X)\big)\Big)$. If $x > y$, algorithm u has produced a partition that matches the ground truth better than algorithm w. Conversely, when $x < y$, algorithm w produced the better match. And when $x = y$, algorithms u and w produced labels that match the ground truth labels equally well. We call this the ARI pairs (ARIP) scheme.

Our experiments use 48 labelled data sets. Any subset of these sets is also labelled, so we can obtain ground truth partitions ($V(X_t)$) for any set X_t of test data. Figures 5 and 6 use the ARIP scheme to compare the efficacy of the four algorithms in this chapter for two tasks: classification with the NCC classifier; and clustering with the k-Means algorithm. The x coordinate in these figures is the ARI match between the ground truth and ShapeSphere (SS) labels. The y coordinate will be the ARI match between the ground truth and the labels produced by one of the three comparison methods; DBA, cDBA or K-Shape.

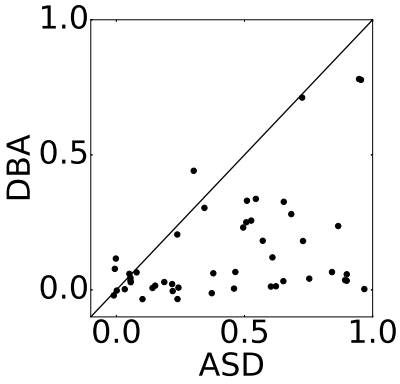
5.6.1 NCC classifier

In our experiments, we used the K-Shape and DBA-Mean implementations available in the K-Shape official library[1] and tslearn[2], respectively. We used the Nearest Centroid Classifier (NCC) for comparing the methods. For each dataset we performed a 5-fold stratified cross-validation and compared the average results over all the folds. In each fold, we calculate a centroid for the training set and then label each testing data point to its closest centroid. For DBA we used DTW, cDBA we used cDTW and for K-Shape (ShapeExtraction) we used SBD to calculate the distance between the centroids and each test member. The algorithms were executed using cython on a one core Virtual Machine with 16GB RAM provided by the NeCTAR cloud[3].

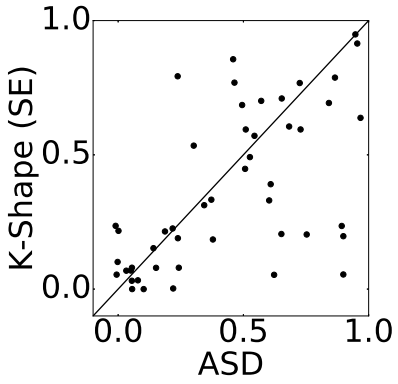
The results of the NCC experiment are shown as an ARIP diagram in Figure 5-5. In Figures 5-5 and 5-6 the line $y = x$ that halves each graph is the line of equality. Any point below this line shows that the winner for that data set is the algorithm on X-axis and vice versa. The graphs show that each of the three methods are superior from others for some data sets and perform poorly for other data sets.The figures show that our method (Contour) outperformed the other methods in classifying the data using the NCC classifier. Figure 5-5d shows the ARIP differences of our method's result to each of the three rivals. The positive median from the boxplot indicates that Contour outperforms all three of the other algorithms for more than half of the datasets in the NCC experiments.

To further analyse these results we first test for normality of the distribution of the ARIP differences. The Shapiro-Wilk test significantly rejected the normality of these distributions ($p << .05$). We conducted a non-parametric Friedman test of differences among the 4 ARI results that rendered a Chi-square= 55.52, which was significant ($p < .05$). The Wilcoxon post-hoc test confirms that the Contour's ARI results in the NCC test are significantly higher than the other methods ($p < .05$); Contour vs SBD: $p = 4e-2$, $T = 432$,

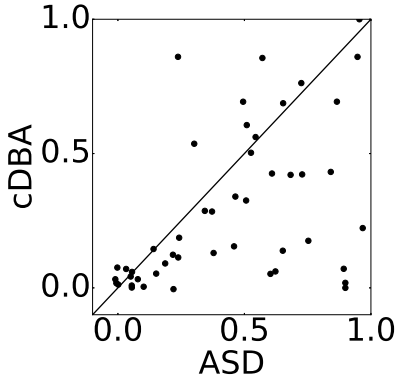
[1] https://github.com/Mic92/kshape
[2] https://tslearn.readthedocs.io
[3] `https://nectar.org.au`



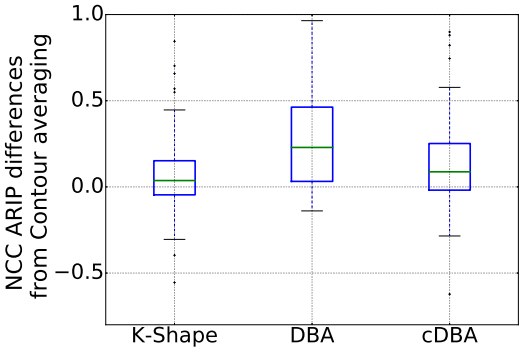
(a) ARIP diagram:ASD vs DBA. ASD performs better in 44 datasets out of 48.



(b) ARIP diagram:ASD vs K-Shape. ASD performs better in 27 datasets out of 48.



(c) ARIP diagram:ASD vs DBA. ASD performs better in 32 datasets out of 48.



(d) Boxplot of differences between ARI results of ASD from other prototype methods. The positive median value shows that ASD performs best for more than half of the data sets.

Figure 5-5: ARIP diagrams and boxplot for the NCC experiments.

Contour vs DBA ($p = 2e-8$, $T = 44$), and Contour vs cDBA ($p = 6e-4$, $T = 254$).

5.6.2 K-Means clustering

We used the K-Shape and DBA-Mean implementations available in the K-Shape official library [4] and tslearn [5], respectively. We modified the tslearn library to support our method. For the cDBA method, despite our multiple efforts to contact the authors, we did not receive any response. We developed cDBA as best we could from the description in the

[4] https://github.com/Mic92/kshape
[5] https://tslearn.readthedocs.io

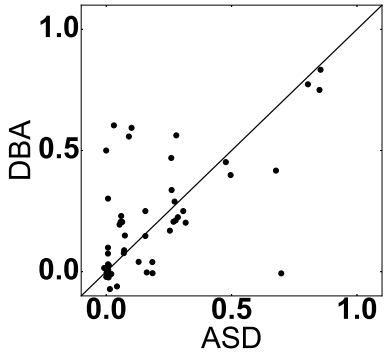
paper. For each dataset we used a 5-fold stratified cross-validation and compared the average over all the folds. For each fold we cluster the dataset using K-Means and validate the clustering result using the ARI. For DBA we used DTW, for cDBA we used cDTW and for ShapeExtraction we used SBD to calculate the similarity distance of each data point to the associated centroid in the K-Means algorithm. All the algorithms were executed in python on an Intel Core i7 desktop machine with 16GB RAM using just one core.

Figure 5-6 shows the ARIP results for our method in comparison with K-Shape (ShapeExtraction), DBA and cDBA. When clustering with K-Means, Contour outperforms K-Shape on the 48 test sets, and performs equal to cDBA and DBA on the 48 test sets (cf. Table 1 in the Appendix), The median values above zero for K-Shape and close to zero for cDBA and DBA in the boxplot in Figure 5-6d support this observation.

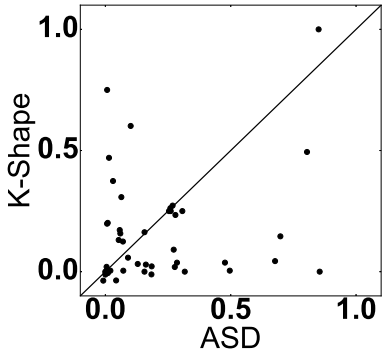
To further understand the mixed results shown in Figure 6, we examined the distribution of the ARIP differences to determine whether the assumption of normality was met. A Shapiro-Wilk test significantly rejected the normality of all these distributions ($p << .05$). A non-parametric Friedman test of differences between the 4 ARI results found them to have nonsignificant differences with Chi-square= 4.92, which suggests a marginally significant result ($p = 0.07$). The Wilcoxon post-hoc test indicated that the ASD's ARI was marginally significantly higher than the K-Shape $T = 415.0$, $p = 0.07$. However no significant difference was observed between ASD vs DBA: $p = 0.76$, $T = 346$ and ASD vs cDBA ($p = 0.78$, $T = 563.0$). This supports our belief that ASD outperforms K-Shape in the K-Means clustering test while performing as well as DBA and cDBA. We discuss this in more detail in Section 5.7.

5.6.3 Complexity

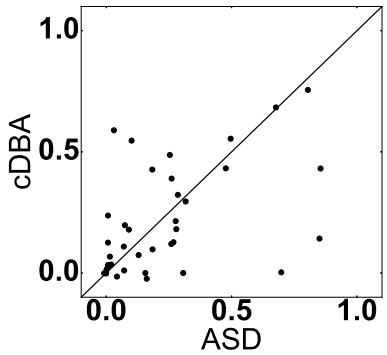
In Section 5.4.1 we proved that calculating the prototype using the Contour algorithm has time complexity $O(mn\log n)$, where m is the number of data points and n is the length of the time series. In Section 5.5.1 we observed that computing the most representative



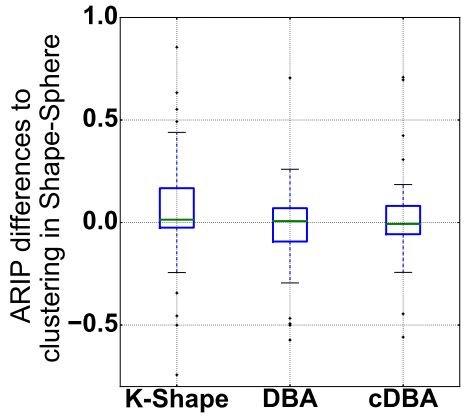
(a) ARIP diagram:ASD vs DBA. ASD performs better in 26 datasets out of 48.



(b) ARIP diagram:ASD vs K-Shape. ASD performs better in 31 datasets out of 48.



(c) ARIP diagram:ASD vs cDBA. ASD performs better in 22 datasets out of 36.



(d) Boxplot of K-Means ARI differences between ASD and the other methods. The ositive median in K-Shape shows ASD performs better compare to this method.

Figure 5-6: ARIP diagrams and boxplot for the K-Means experiments.

prototype using DBA (as well as cDBA) is an NP-complete problem. Computing prototypes using K-Shape is $\Omega(n^2 + m^2)$ in time, where Ω is the lower bound. Thus theoretically our algorithm is the most efficient algorithm of the four discussed here. Since the most time-consuming task in both NCC and K-Means clustering for waveform analysis is to calculate the prototypes, we empirically analyse the four algorithms for computing a prototype from a set.

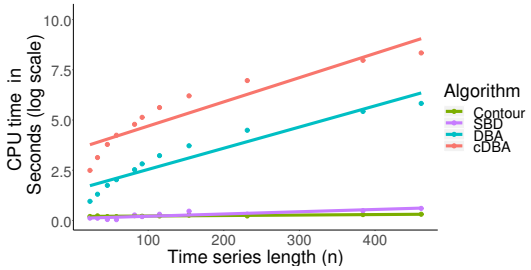
Figure 5-7 shows the CPU times of different algorithms for computing prototypes by changing the two main input parameters, namely, the length of the time series n and the number of data points m. Figure 5-7a and 5-7c show the effect of changes in the length

of time series data points on the performance of each algorithm. In this test, we used a set of time series with 98 data points (OSULeaf dataset [171]). For each test we resample each time series to get the required length.

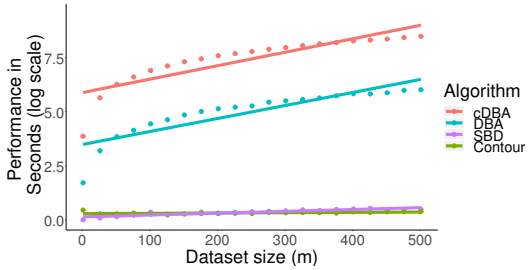
To analyse the effect of n on the performance of each algorithm we performed an analysis of variance. The Bartlett's test result showed a violation of homogeneity of the variances (Chi-square = 356, $p = 2e-16$). Therefore, we used the non-parametric Kruskal-Wallis test that revealed significant differences between CPU times as a function of time series length (Chi-square= 33.36 and $p < 0.05$). A post-hoc Wilcoxon reveals the significant differences between the performance of Contour and DBA ($p < 0.05$, $r = 0.88$) and Contour and cDBA ($p < 0.05$, $r = 0.89$). However, no significant difference was found between the performance of Contour and SBD ($p = 0.7$).

Figures 5-7b and 5-7d show the effect of changes in the number of data points m on the performance of each algorithm. In this test we randomly sampled data points from the ECG5000 dataset [171]. To analyse the effect of m on the performance of each algorithm we performed an analysis of variance. The Bartlett's test result showed a violation of homogeneity of the variances (Chi-square = 726.47, $p = 2e-16$). Therefore, we used the non-parametric Kruskal-Wallis test that revealed significant differences between CPU time as a function of dataset size (Chi-square= 69.07 and $p < 0.05$). A post-hoc Wilcoxon test reveals significant differences between the performance of Contour and DBA ($p < 0.05$, $r = 1.24$) and Contour and cDBA ($p < 0.05$, $r = 1.25$). The test shows a marginally significant difference between the performance of Contour and SBD ($p = 0.09, r = 0.35$).

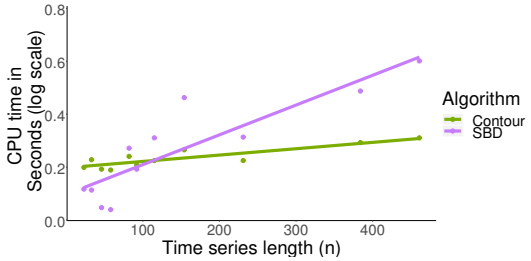
As a reference, the average cross-validation time for both the NCC and clustering scenarios are shown in Figure 5-8. As explored and expected, Contour and SBD are the most efficient algorithms in both NCC and K-Means clustering.



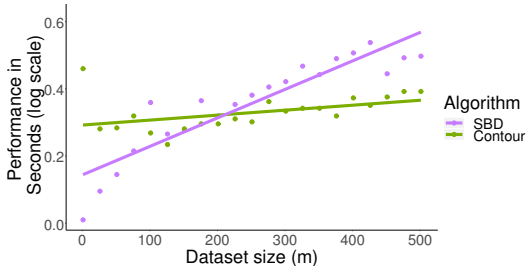
(a) CPU time as a function of time series length.



(b) CPU time as a function of data set size.

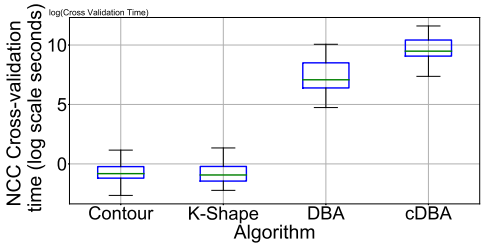


(c) CPU time of SBD and Contour algorithms for computing prototypes as a function of time series length.

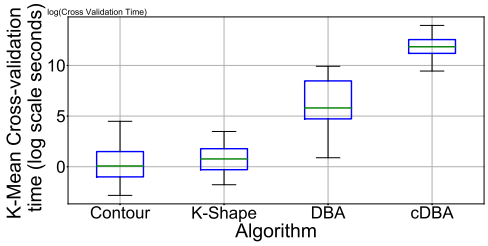


(d) CPU time of SBD and Contour algorithms for computing prototypes as a function of data set size.

Figure 5-7: CPU times the four algorithms for computing a prototype from a single set of data points.



(a) Average cross-validation time in NCC experiment.



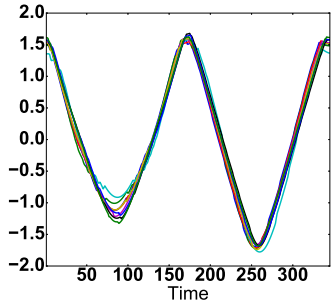
(b) Average cross-validation time in K-Means clustering experiment.

Figure 5-8: Average fitting time (log-scale) for each algorithm in both experiments.

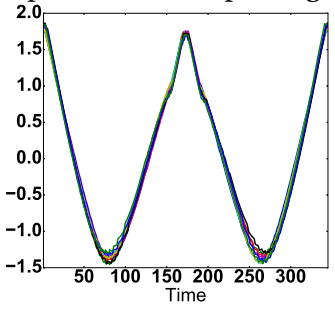
5.7 Discussion

The experiments in Section 5.6 show that the proposed methods based on Shape-Sphere for analysing a finite set of time series outperforms the three comparison methods in NCC classification for the 48 data sets. Our clustering experiment showed the proposed method performs at least as well as the three comparison methods. For comprehensive details of each method see Table 5.1 in the Appendix.

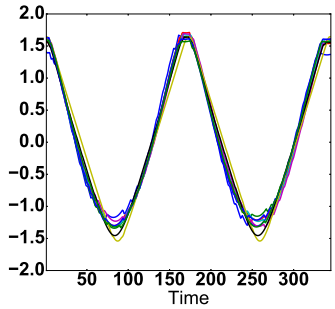
A closer look at the data sets reveals where each method performs well while other methods fail. First, Shape-Series accurately extracts the overall shape of the time series. ASD amplifies the shape differences while averaging these changes in the other methods levels small changes between groups. For example, consider the Diatom Size Reduction dataset, which contains four classes (Figure 5-9) [171]. Although the overall look of each class is the same (they all start with a trough follows by a peak which is followed by another trough), the shape of each class is different in terms of the shape of its peak and troughs. For example, class 1 has a shallow trough follows by a flat peak and a deep trough.



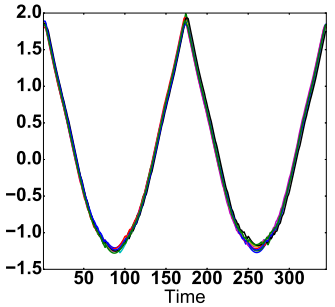
(a) Members of class label 1.



(b) Members of class label 2.



(c) Members of class label 3.

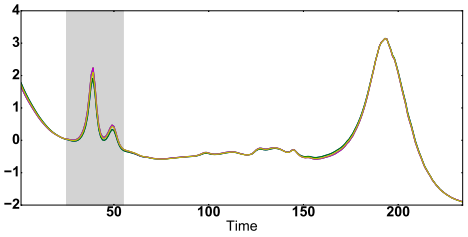


(d) Members of class label 4.

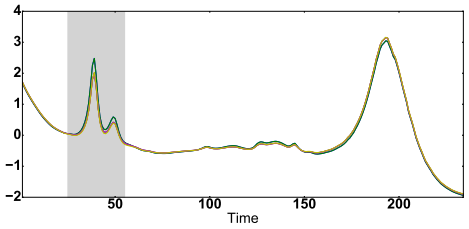
Figure 5-9: The four classes in Diatom Size Reduction data set. Geometric shape is a distinguishing feature.

Our method fails to recognise clusters when the shape is not the differentiating feature. For example, consider the time series for the two classes in the Wine data set, shown in Figure 5-10 [171]. From the figure, we can see that the shapes of the time series are identical.

Second, K-Shape computes cross-correlation of the signals. As a result, it is biased towards high and low value peaks of the data sets. K-Shape can identify differences in data



(a) Members of class label 1 from Wine data set drawn on top of each other. The distinguishing region of the time series is highlighted



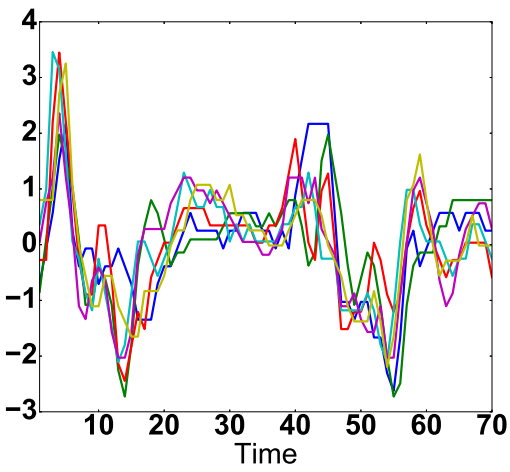
(b) Members of class label 2 from Wine data set drawn on top of each other. The distinguishing region of the time series is highlighted.

Figure 5-10: The two classes in Wine data set. Geometric shape is not a distinguishing feature of the subsets.

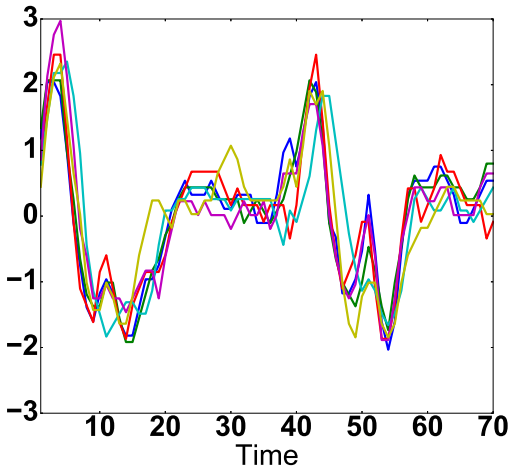
sets where peaks are the descriptive characteristics of the data. For example consider the Wine data set in Figure 5-10. In this figure the shapes of the two waveforms are essentially identical, the difference being the height of the first peak.

Third, DTW works best wherever sub-regions of the time series must align with each other. In this scenario, DTW can reduce the effect of the sub-regions that poorly match each other by warping them to the best possible sub-region (to alleviate the mismatch penalty) while matching the aligned sub-regions. For example, consider the Sony AI Robot data set with 2 classes (Figure 5-11) [173]. In this data set, neither shape nor a sub-region are distinguishable features. Thus, the alignment of time series with each other is an essential step in analysing the data.

Finally, the main reason Shape-Sphere analysis cannot consistently outperform the other methods for the K-Means clustering of time series is due to how the average shape of two different shapes can be a different shape (for example, averaging a square and a circle creates a square with rounded corners). Therefore, at any iteration of K-Means, it is possible that a few items from different classes are mixed together this way, which in turn results in a prototype that attempts to represent both classes rather than just one class. One solution for this problem is to have more data samples so that the effect of a few incorrect items is reduced while averaging the Shape-Series. However, Shape-Series outperforms the other methods significantly (NCC test case) by avoiding mixing items of



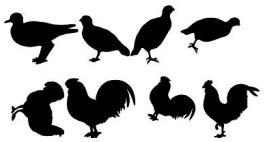
(a) Members of class label 1.



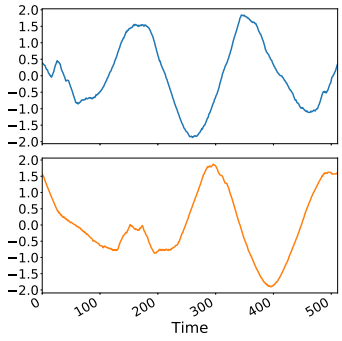
(b) Members of class label 2.

Figure 5-11: The two classes in Sony AI Robot data set. Alignment of waveforms is an essential part in comparing them.

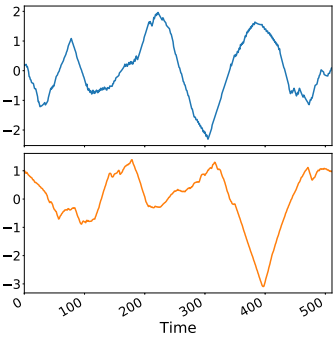
different classes.



(a) Samples of images from BirdChicken data set.



(b) Class label 1. Time series are scaled and shift rotated.



(c) Class label 2. Time series are shift rotated to right.

Figure 5-12: Samples of images and time series from BirdChicken data set.

Our next example concerns the BirdChicken data set. This data set is produced from an image data set of chickens and other birds. For each image, the outline of the image is extracted and mapped into a 1-D series of distances to the centre. Examples of images and time series data from this data base are shown in Figure 5-12. This data set has the characteristic that images can be rotated or scaled so we can analyse the impact of these transformations on the sampled time series. For example, the global minimum of the bottom time series in both sample classes shown in Figure 5-12b and Figure 5-12c are shifted to the right compared to the top time series of the same class. The scale and

rotational invariance properties of our method make it superior to the other methods in our experiments in both classifying and clustering this data set (see the results in Table 5.1, Supplementary Material).

5.8 Conclusions

In this chapter, we proposed Shape-Sphere, a pseudometric space for representing shape features of time series. We showed how to efficiently transform time series into Shape-Sphere. We proved that Shape-Sphere is a pseudometric space and showed how to use it to compute the most efficient shape average for a given cluster of time series. We proved theoretically and showed experimentally that the angular distance in Shape-Sphere (ASD) has the best time complexity of the four methods tested. On average, ASD yielded the most accurate labelling results when used for NCC classification where shape is a differentiating factor. The results of the K-Means test were mixed; ASD labelling was better for some data sets, but not so good for others. Our analysis by cases suggests that, as always, the data themselves often dictate which of the methods discussed will be most useful for clustering. With unlabelled data (the real situation in cluster analysis), there is no easy way to decide which of the situations may apply. Consequently, an ensemble approach using all four methods [174] might be the best way to ensure reliable results. Beyond what we showed here, since our time series transformation is a one-to-one function, the transformed results can be used in a variety of time series analysis tasks from classification to indexing strategies.

5.9 Appendix

5.9.1 The impact of noise

We analyse the impact of six different common noise types on our algorithm and the state-of-the-art algorithms. For our analysis we have used five common types of coloured noise plus missing-value noise. We analyse the impact of each noise type on each algorithm. We then discuss how well each algorithm performs in the presence of noise.

Method

To perform the analysis, we selected the Plane dataset from the UCR data, which most algorithms can cluster well (ARS > 0.5). To analyse the impact of each type of noise on the different metrics, we first select a noise type to add to every member of the data set. We then apply a smoothing filter (a moving average filter) as a preprocessing method that can be used in practice. We perform 5-fold-cross validation, i.e., we divide the Plane dataset into five different subsets such that the intersection of each pair of subsets is empty. For each fold one subset leaves out and apply k-means on the rest of the data set. We test the learnt model on the left out subset by assigning each member of the subset to the cluster with the closest centroid to the member and calculate the ARS. The average ARS over the five test subsets is reported.

Coloured noise

Coloured noise corresponds to additive noise that can be present in the data. Common types of coloured noise are: (1) White noise: a noise signal that has equal density at different frequencies. (2) Blue noise: a noise signal whose power density increases by 3dB per octave with increasing frequency. (3) Violet noise: a noise signal whose power density increases by 6dB per octave with increasing frequency. (4) Red noise: a noise signal whose power density decreases by 6dB per octave with increasing frequency. (5) Pink noise: a

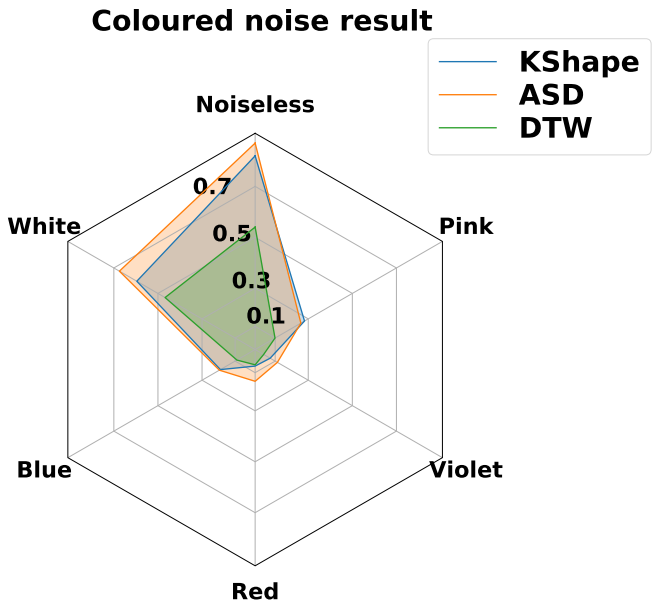
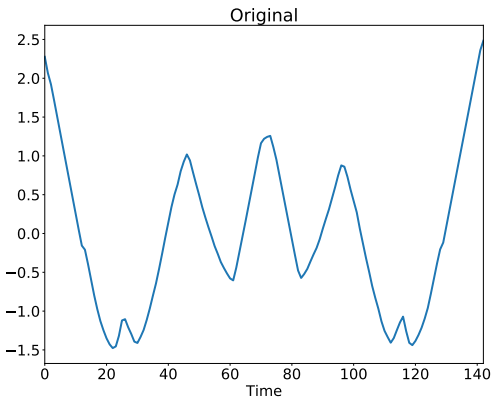


Figure 5-13: K-Means ARS result for different algorithms after adding various types of coloured noise to the Plane data set.

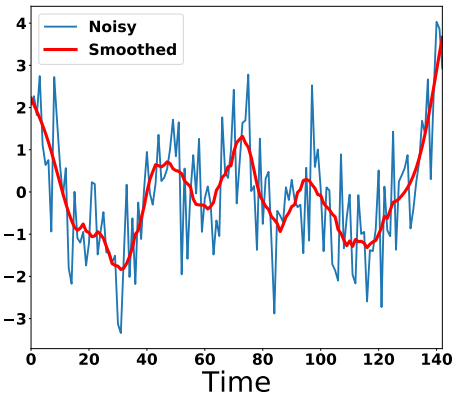
noise signal whose power density decreases by 3dB per octave with increasing frequency.

The result of each algorithm is shown using a spider chart in Figure 5-13. Each line in the chart presents an ARS level with the centre of the chart representing ARS equal to zero. Thus, the further from the centre, the better the algorithm has performed. The result of each algorithm is shown in a different colour. Although the performance of all the algorithms has dropped in comparison to the noiseless data sets, we can see that all the algorithms performed well in the presence of white noise. This is because the effect of white noise can be reduced using a smoothing filter. The average ARS for all the algorithms dropped as a result of adding other types of coloured noise. However, our method (ARS) outperformed the other methods in clustering the Plane data set in the presence of white, red and violet noise. Our algorithm and K-Shape have similar performance in the presence of blue noise and K-Shape outperforms the other methods in the presence of pink noise.

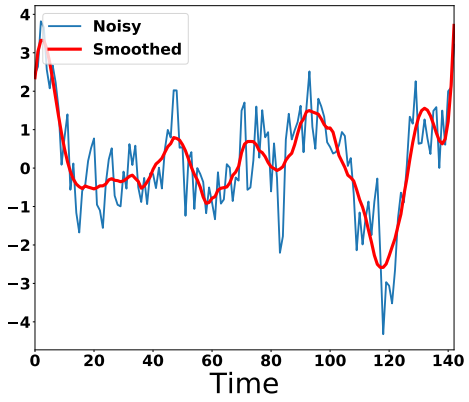
We show a sample signal from the Plane data set together with the effect of each type of noise and its smoothed version of the signal in Figure 5-14. As expected white noise has



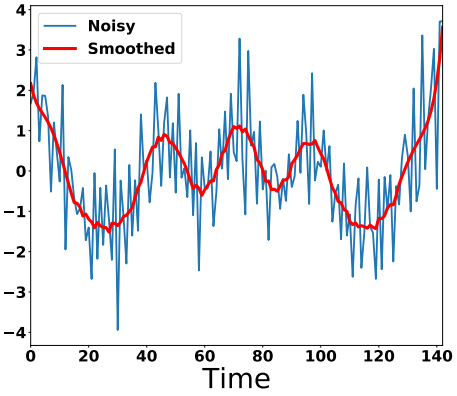
(a) The original time series from Plane data set.



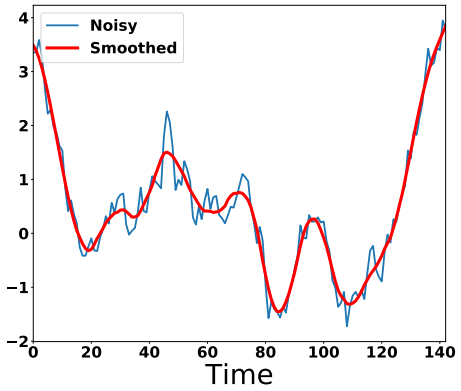
(b) The effect of white noise.



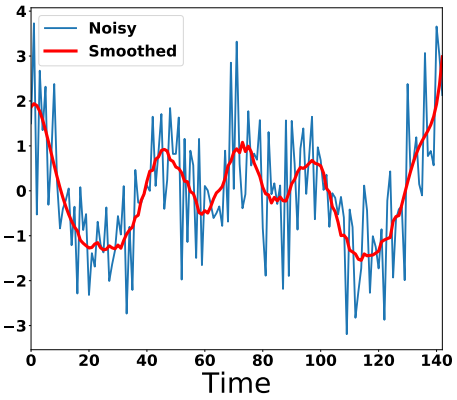
(c) The effect of pink noise.



(d) The effect of violet noise.



(e) The effect of red noise.



(f) The effect of blue noise.

Figure 5-14: The effect of different coloured noise types on a sample data point from the Plane data set.

the least effect on the shape of the signal. A smoothing filter can substantially reduce the white noise from the signal. However, coloured noise signals towards the low frequencies distort the original signal the most, which reduces the accuracy of K-Means clustering.

Missing samples

The second type of noise that often impacts time series analysis is missing samples. To analyse the impact of missing samples we use the Plane data set discussed in the previous section. From each member, we select n random points using a uniform distribution and set them to zero. We then apply K-Means clustering as described in the method section. The results of K-Means can be seen in Figure 5-15. The figure shows that our method

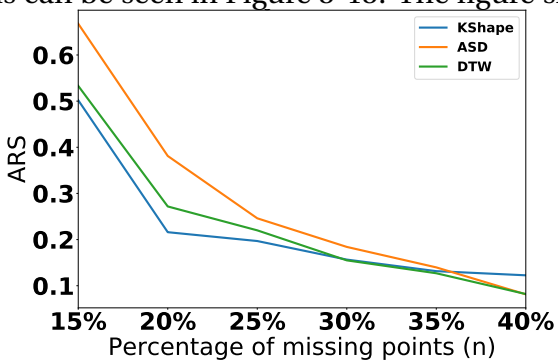


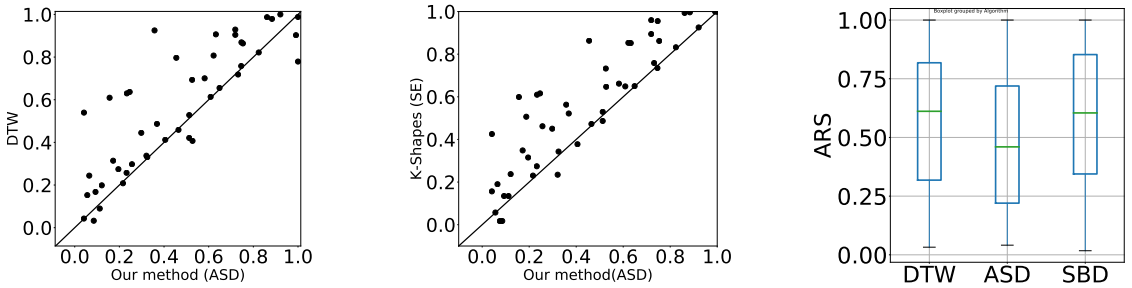
Figure 5-15: K-Means ARS result for different algorithms on the Plane dataset with missing samples where n is the percentage of missing samples from each item in the data set.

(ASD) outperforms other methods when the number of missing samples are less than 35% of the length of the time series. We can see that the performance of all the methods decreases as the number of missing samples increases. However, K-Shape's performance is more stable when the percentage of missing samples exceeds 25% of the length of the time series, while the other methods' performance still decreases.

5.9.2 One Nearest Neighbour classifier

It has been shown that the One Nearest Neighbour (1NN) classifier is one of the most accurate classifiers for time series data sets [175, 176]. In 1NN each unlabelled data instance is assigned to the class of the closest item from the training data set. We applied

the ASD similarity measure using the 1NN classifier. Figure 5-16 shows a comparison of SBD and DTW to ASD for 1NN classifiers using the 48 data sets from Table 5.1. As



(a) DTW versus ASD. (b) SBD versus ASD. (c) Comparison of 1NN results.

Figure 5-16: Comparison of ARS results of 1NN classifier using different distance/similarity measures.

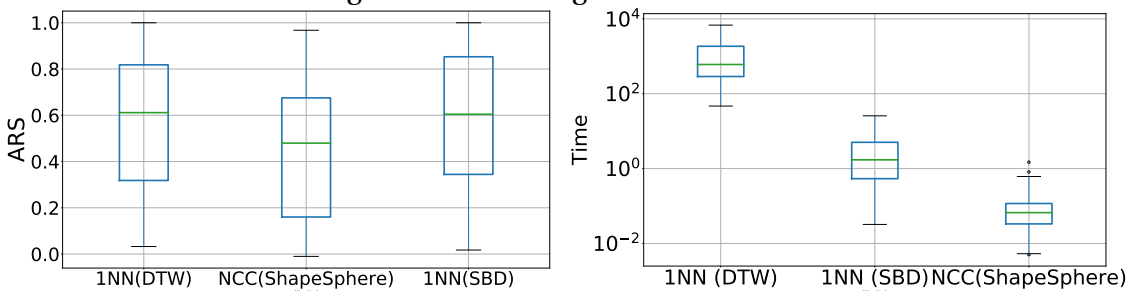
shown in Figure 5-16, for 30% of the data sets in our experiments both DTW and SBD outperform our method ASD, resulting in the ARS average difference of DTW from ASD to be 7% (quantiles are at 0.001, 0.05 and 0.16) and SBD to ASD to be 9% (quantiles are at 0.007, 0.09, 0.19). This loss in performance is because the angular distance is inelastic in the pair-wise comparison of time series, that is, unlike DTW and SBD, the vector representation of the time series in the Shape-Sphere space is fixed. The elasticity property is an important property for 1NN classifiers where a pair of time series evolve to find the best match between them when calculating the similarity of the pair as shown by Xi et al. [175]. This loss of elasticity impacts our 1NN results, where for some data sets both DTW and SBD perform better than ASD. Note that a possible direction for future research is to use SBD or DTW with the Shape-Series representation.

5.9.3 Comparing One Nearest Neighbour to Nearest Centroid Classifier

We demonstrate that the ASD method in combination with the Contour prototype is the most effective technique in classifying time series using a Nearest Centroid Classifier. In Section 5.9.2 we compared different distance/similarity methods using a one nearest

neighbour (1NN) classifier. However, a challenge with using 1NN to classify a new item is its computation time.

Figure 5-17b shows a comparison of the CPU time for classifying time series using 1NN (SBD and DTW) and NCC (ASD). The figure shows (on a logarithmic scale) that 1NN is much slower than NCC. The remaining question is: Do we lose any accuracy for this efficiency gain? Comparing the NCC results to 1NN in Figure 5-17a shows that NCC on average performs worse than 1NN (average ARS difference $\mu = 12$ with standard deviation $\sigma = 0.31$). However, having a large standard deviation ($\sigma > 2 * \mu$) confirms that there are situations where NCC outperforms 1NN (15 data sets out of 48 compared to 1NN (SBD) and 13 data sets out of 48 compared to 1NN (DTW)) while it is faster than 1NN (Figure 5-17b). Figure 5-17b compares the CPU time for the 1NN algorithms to NCC using our method (Shape-Sphere). Note that the efficiency gain of our algorithm compared to 1NN is several orders of magnitude on the logarithmic axis.



(a) Classification result (ARS) comparison

(b) Classification time comparison. The average classification time for each algorithm is shown next to its box.

Figure 5-17: Comparison of 1NN classifier through SBD and DTW to NCC classifier using Shape-Sphere.

5.10 Experimental results

The results of our NCC (classification) and K-Means (clustering) tests are given in Table 5.1, which contains the (average) values of the adjusted Rand index comparing the ground truth partition to a computed one on each of the 48 data sets. For both scenarios the winning algorithm results are in bold. In Section 5.6.3 we showed that cDBA is the slowest

algorithm for computing the prototype, which was also reported by the authors. Due to resource limitations in terms of computation time, we could not obtain the results for all the reported datasets using cDBA in our clustering scenario. In these cases, we show these results as NA.

Table 5.1: ARIs for our experiments using NCC and K-Means.

	Dataset	NCC (Classification)				K-Means (Clustering)			
		ASD	**KS**	**DBA**	**cDBA**	**ASD**	**KS**	**DBA**	**cDBA**
1	DiatomSizeReduction	**0.95**	**0.95**	0.78	0.86	**0.86**	0.00	0.83	0.43
2	BirdChicken	**0.90**	0.20	0.06	0.01	**0.31**	0.25	0.25	0.00
3	MoteStrain	**0.73**	0.59	0.18	0.42	0.07	-0.16	0.15	**0.20**
4	MiddlePhalanxTW	0.54	**0.57**	0.34	0.56	0.50	0.00	0.40	**0.55**
5	ProximalPhalanxTW	0.49	0.69	0.23	**0.7**	**0.48**	0.04	0.45	0.43
6	Plane	0.95	0.91	0.78	**1.0**	**0.81**	0.49	0.77	0.76
7	Car	**0.89**	0.24	0.04	0.07	**0.19**	0.02	-0.01	0.10
8	Adiac	**0.34**	0.31	0.30	.29	**0.28**	0.02	0.21	0.21
9	ItalyPowerDemand	**0.97**	0.64	0.00	0.22	**0.70**	0.15	-0.01	0.00
10	DistalPhalanxOutlineCorrect	**0.62**	0.05	0.01	0.06	**0.01**	0.01	0.03	0.03
11	WordsSynonyms	**0.65**	0.21	0.03	0.14	**0.16**	**0.16**	0.15	NA[6]
12	FISH	**0.61**	0.39	0.12	0.43	0.29	0.04	0.22	**0.32**
13	SwedishLeaf	**0.51**	0.45	0.25	0.33	0.27	0.09	**0.29**	NA
14	FaceFour	**0.68**	0.61	0.28	0.32	0.26	0.26	**0.47**	NA
15	MidPhalanxOutlineAgeGroup	**0.53**	0.49	0.26	0.50	0.19	-0.01	0.04	**0.42**
16	Computers	**0.90**	0.05	0.03	0.02	**0.02**	0.00	0.01	NA
17	WormsTwoClass	**0.22**	0.01	0.00	0.00	**0.01**	-0.01	-0.02	NA
18	ECG200	**0.75**	0.20	0.04	0.18	**0.29**	0.27	0.21	0.13
19	CBF	**0.87**	0.79	0.24	0.70	0.18	0.37	**0.60**	0.59
20	TwoLeadECG	**0.84**	0.69	0.07	0.43	0.04	**0.20**	0.10	0.13
21	Cricket_Y	**0.38**	0.18	0.06	0.13	0.06	0.17	**0.21**	NA
22	SonyAIBORobotSurfaceII	**0.37**	0.33	-0.01	0.28	0.29	0.23	**0.56**	0.18
23	OSULeaf	**0.60**	0.33	0.01	0.05	0.07	**0.31**	0.21	NA

[6]Not Available. See the Appendix for details.

24	PhalangesOutlinesCorrect	**0.15**	0.08	0.02	0.05	0.02	0.00	-0.01	**0.03**
25	ToeSegmentation1	**0.24**	0.08	0.01	0.18	0.08	**0.20**	-0.02	0.03
26	BeetleFly	**0.24**	0.19	-0.03	0.11	0.16	0.00	**0.25**	0.00
27	Worms	**0.08**	0.03	0.06	0.03	0.01	-0.01	**0.03**	NA
28	Herring	**0.10**	0.00	-0.03	0.01	-0.00	-0.04	**0.02**	0.00
29	Earthquakes	**0.06**	0.00	0.03	**0.06**	0.00	**0.02**	-0.01	NA
30	MiddlePhalanxOutlineCorrect	**0.05**	0.03	0.05	0.01	0.02	0.00	**0.07**	0.03
31	DistalPhalanxTW	0.72	**0.77**	0.71	0.76	**0.68**	0.04	0.42	**0.68**
32	ProPhalanxOutlineAgeGroup	0.51	0.59	0.33	**0.60**	**0.32**	0.00	0.20	0.12
33	DistalPhalanxOutlineAgeGroup	0.30	0.53	0.44	**0.54**	0.27	0.25	**0.34**	0.03
34	Gun_Point	0.00	**0.22**	0.00	0.10	**0.05**	-0.04	-0.06	-0.01
35	synthetic_control	0.57	0.70	0.18	**0.85**	0.10	**0.60**	0.59	0.54
36	Coffee	0.24	0.79	0.21	**0.86**	0.86	**1.00**	0.75	0.15
37	ECGFiveDays	0.46	**0.86**	0.00	0.15	0.01	**0.47**	-0.07	0.07
38	Wine	0.05	**0.08**	0.04	0.01	0.01	**0.75**	0.30	0.24
39	SonyAIBORobotSurface	0.46	**0.77**	0.07	0.40	0.10	0.06	**0.56**	0.18
40	ProPhalanxOutlineCorrect	0.14	**0.15**	0.01	0.14	0.10	**0.12**	0.08	0.11
41	OliveOil	0.65	**0.71**	0.33	0.69	0.26	0.25	0.17	**0.49**
42	Cricket_Z	0.19	**0.21**	0.03	0.10	0.08	0.13	**0.19**	NA
43	Cricket_X	0.22	**0.23**	0.02	0.12	0.07	0.16	**0.23**	NA
44	ShapeletSim	-0.01	**0.24**	-0.02	0.04	0.01	0.00	**0.50**	0.00
45	Beef	0.00	0.10	**0.12**	0.08	**0.17**	0.03	0.00	-0.02
46	Strawberry	0.05	**0.07**	0.06	0.04	0.08	0.00	**0.09**	0.01
47	Ham	0.03	0.07	0.00	**0.08**	**0.01**	0.00	0.00	**0.01**
48	Lighting2	-0.01	0.05	**0.08**	0.02	**0.13**	0.03	0.04	NA

Chapter 6

LiftSmart: The wearable that monitors weight training activities in real-time, without the use of labelled data

6.1 Introduction

In this chapter, we demonstrate LiftSmart, a novel smart wearable to detect, track and analyse weight training activities. To the best of our knowledge, LiftSmart is the first wearable for weight training that is based on unsupervised machine learning techniques, specifically those introduced in the previous chapters, to eliminate the use of labelled data. In Chapter 1 we discussed the importance of methods that can adapt to personalised routines for monitoring sport activities. Chapter 3 followed this idea of personalised monitoring where we designed a novel workflow to detect incorrect performance of an exercise by solely observing correct performance of an exercise. Chapters 4 and 5 were the building blocks of designing resource efficient machine learning techniques that can run on a resource limited environment like wearable devices. In this chapter, we combine all our previous efforts in designing resource efficient machine learning techniques to

build LiftSmart—the first wearable that monitors weight training activities without the use of any labelled data in real-time.

6.2 LiftSmart

LiftSmart is an unsupervised wearable solution that, independently of any other computing resources, addresses challenges related to the use of labelled data that supervised approaches face. It starts tracking an exercise set from the incoming data stream on the wearable device by using the OToR algorithm (described in Chapter 4).

After identifying the first three repetitions of an exercise, the wearable computes the observed prototype from the first identified repetition using the Contour Algorithm described in Chapter 5, and considers it as the correct performance of the exercise. LiftSmart compares the new incoming repetitions to the computed prototype through their shape using the ASD method described in Chapter 5. The comparison between each new repetition and the first repetition results in a score, called $Dist$, between 0 and 1 with zero being an exact match and one being completely different. We use this value to provide feedback to the athlete while performing the exercise. Although configurable, we have considered every 0.2 change in the $Dist$ score to be a shift from the level of consistency. Note that setting the shift value to be too strict, results in repetitions needing to be exactly the same, whereas having a larger shift (> 0.2) results in having more degrees of freedom to capture inconsistencies between repetitions. We use a 5-level consistency system (1) Accurate (A) ($Dist \leq .2$), (2) Close (C) ($.2 < Dist \leq .4$), (3) Deviating (D) ($.4 < Dist \leq .6$), (4) Fluctuating (F) ($.6 < Dist \leq .8$), and (5) Incomparable (I) ($.8 < Dist$) to give feedback to the athlete. The computed prototype is considered as the reference repetition to evaluate the consistency of other repetitions in real-time.

For offline processing, a mobile application stores the detected exercise signals. LiftSmart provides various methods to compare repetitions with a reference signal, such

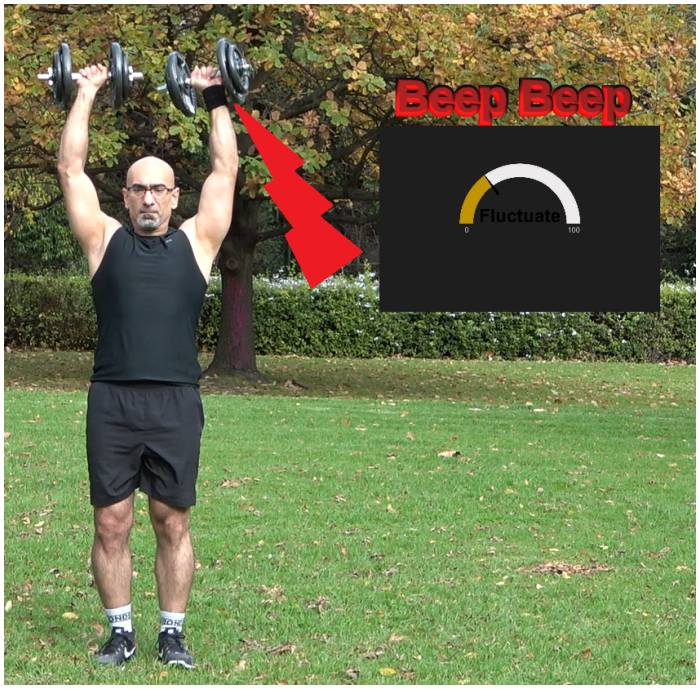


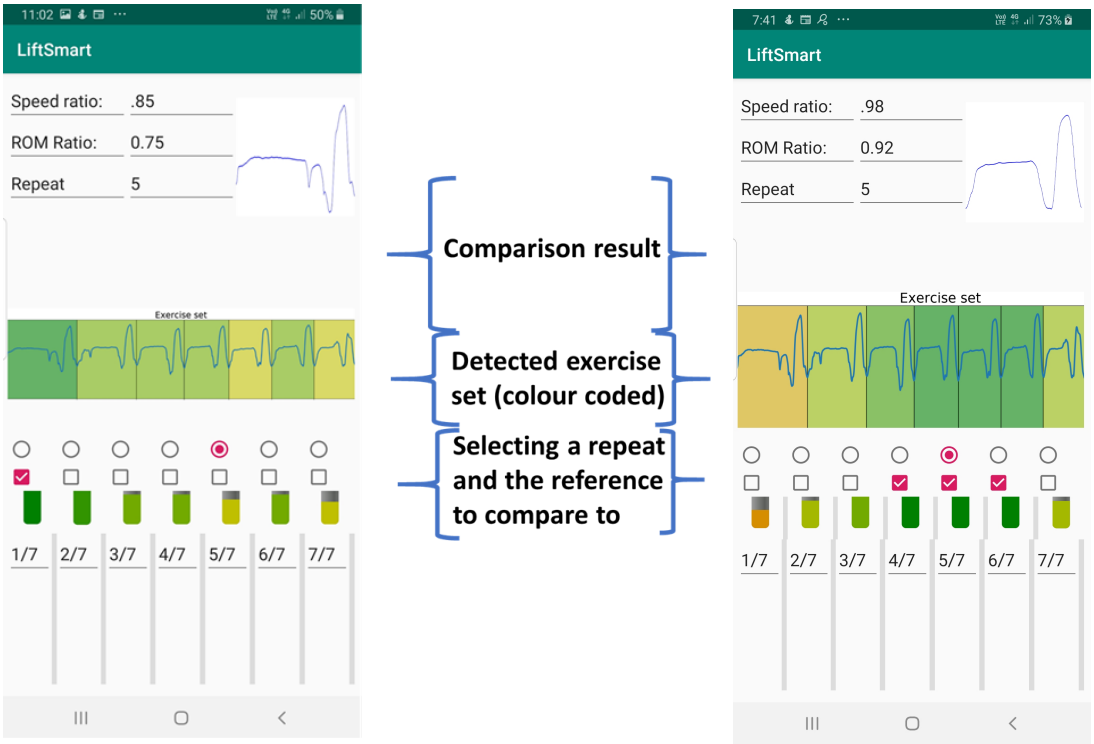
Figure 6-1: The wearable monitors and evaluates the performance in real-time. It warns the user when deviating from the correct posture.

as selecting a single repetition or a set of repetitions as a reference signal. An interactive application shows each repetition of the exercise to the athletes and their coaches (Figure 6-2).

6.3 Method

The tasks of detecting, tracking and analysing weight training performance are based on analysing the incoming stream of data captured by the gyroscope on the wearable device.

The workflow (designed in Chapter 3) detects and analyses repeating exercises by leveraging the consecutive repeating property of weight training exercises [177]. A consecutive repeating movement manifests itself as a consecutive repeating signal in the incoming stream of data. Detecting these consecutive repetitions allows us to detect when the trainee started an exercise. It also allows us to find the repeating signal as the reference signal. After selecting a reference signal, we can search for new repetitions from



(a) (b)

Figure 6-2: Two applications of LiftSmart in offline mode. The application shows each repetition in a colour coded manner. The colour of the repetition results from its comparison to the reference. (a) Trainees can use the application to set the reference repetition and compare the other repetitions to the reference repetition. The application shows the comparison of the repetitions (the speed ratio and Range of Motion (ROM) ratio). (b) Trainers and expert users can select multiple repetitions of the same set using TickBoxes to analyse the performance of the set. By selecting multiple repetitions, any anomalies can be found.

the incoming data that are (1) consecutive to the last detected signal and (2) similar to the reference signal. The workflow for detecting and tracking and evaluating weight training movements using the incoming data stream is presented in Figure 6-3.

6.4 Application

Detecting, tracking and analysing weight training exercises in real-time and on-line provides an opportunity for improving trainees' outcomes from the training routines while enhancing the trainee-trainer communication cycle. We discuss two of the main

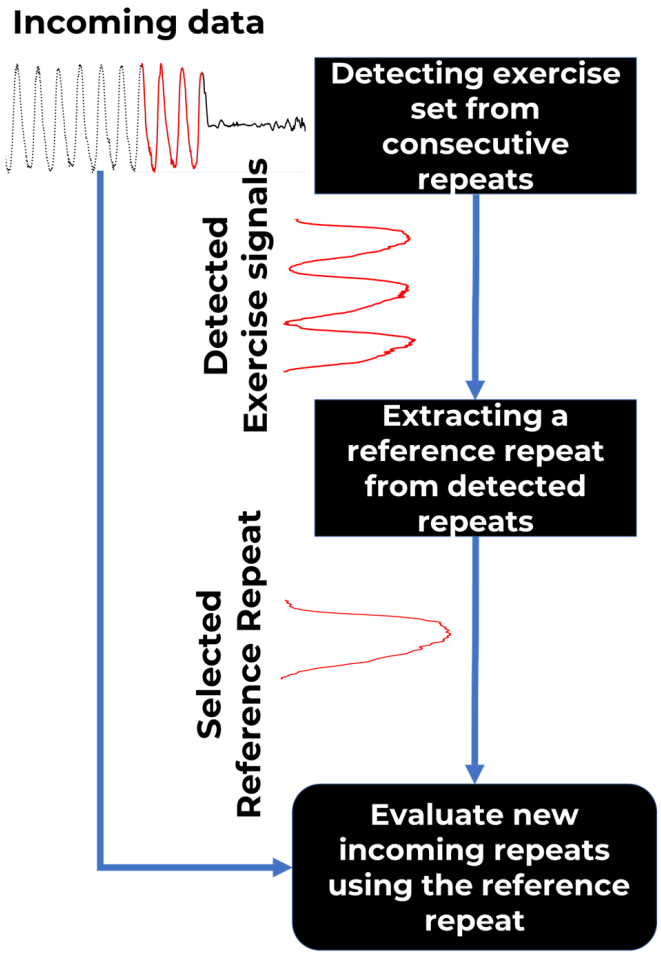


Figure 6-3: The workflow for unsupervised evaluation of a weight training exercise. First, a detection method is required to detect the exercise from the incoming data stream. Second, a repetition from the detected exercise set is selected as the reference set. Third, each new repetition of the same set is evaluated using the selected signal of repeat.

opportunities in this regard.

6.4.1 Feedback to athletes while training

The first part of LiftSmart provides a warning mechanism to experienced athletes in regards to muscle fatigue—the main cause for many injuries. If athletes are familiar with an exercise, they can perform it accurately under light load. However, by increasing the load, athletes often push themselves too hard. Too much pressure over an extended

training period or sudden jumps in the load can cause muscle fatigue, which manifests itself through significant deviations after a few repetitions. By comparing the repetitions to a correct execution, we ensure that the athlete's training is consistent throughout the set and can warn them when they deviate from the reference by more than a configurable threshold (see Figure 6-1).

6.4.2 Offline analysis of trainee's performance

The second part of LiftSmart provides a mechanism for feedback about the athlete's performance offline. Coaches can use LiftSmart remotely. Historically, athletes work with their coaches for one or two sessions and then start practising the exercise for a few weeks before meeting their coaches again. In this process the coach have traditionally relied on the athlete's memory to perform the exercises up to a certain standard. LiftSmart can provide meaningful feedback about the athletes' performance during training to the athletes and their coaches online as well offline. Coaches can use the system to monitor the athlete's performance and update their exercise routines accordingly (see Figure 6-2).

6.4.3 Demo

A demo of LiftSmart is available at `https://www.youtube.com/watch?v=R3saws37KKM`

6.5 Conclusions

In this chapter, we demonstrated the capability of the machine learning techniques that have been designed in this thesis to detect and track weight training exercises online on a wearable device in real-time. The workow is based on detecting short bursts of a repetitive signal from the incoming data stream, extracting a reference signal and using the reference signal to evaluate the new repetitions presented in Figure 6-3. The main use case behind our scenario is based on the trainees' knowledge in performing the exercise

correctly but cannot maintain the correct form due to pressure and/or muscle fatigue, as was discussed in Chapter 1. By making use of the trainee's knowledge of how to perform an exercise, signals from the early stage of a detected set (the rst three repetition signals) can be used to derive a prototype from these signals as a reference to evaluate any new signals. The selection of the rst three signals of repetition for an exercise is the result of an observation that the rst few repetitions are often accurate repetitions of an exercise that a trainee performs during their routine. Athletes often deviate from the correct posture after performing the rst few repetitions as they become fatigued due to the weight load they are holding during their routine. However, further investigation is required to identify the most accurate method for selecting the optimum number of repetitions when computing a reference signal for evaluation of incoming signals. Note that LiftSmart provides the potential for further HCI research in designing a trainee/trainer review cycle where trainers can tune the correctness of repetitions to set when LiftSmart should warn the trainee. It generates a new environment that can bring remote/offline monitoring into the cycle of trainees/trainers interactions.

LiftSmart is sensitive to the position of the wearable, that is, if the wearable is worn on a body part that does not show any repeating pattern then the system fails to provide any feedback to the trainees and trainers. A direct extension for LiftSmart is to design a sensor fusion platform where multiple instances of the wearable sensor are worn by a trainee, and feedback is generated by aggregating the output of each wearable sensor.

Lastly, LiftSmart provides its feedback in a visual output format. However, since the processing takes place on the wearable, it is feasible to generate other types of warning signals to the trainee when a deviation from their normal performance is detected. For example, audio messages can be sent to a headphone worn by the athlete, or device vibration can be used to warn the trainee. Selecting the best mechanism to provide this warning feedback to the trainees and trainers warrants further investigation.

Chapter 7

Future work and Conclusions

7.1 Conclusions

In this thesis we have addressed the problem of monitoring weight training activities in real-time using a wearable device without the use of labelled training data. We have outlined the importance of this real-time monitoring task and the machine learning challenges that arise for solving this problem. We have investigated the current state-of-the-art techniques and discussed in detail their shortcomings in answering the challenges for unsupervised monitoring of weight training activities in real-time over a wearable device. In particular, we have investigated three main challenges that the current methods have failed to address in a unified manner, namely:

1. Personalised monitoring,
2. Real-time monitoring, and
3. Wearable-based monitoring.

To answer each of these challenges we have designed novel, resource-efficient machine learning analysis techniques that can run on wearable devices with limited computing resources. We summarise our contributions in this thesis in this section.

In Chapter 3, we designed a system that detects incorrect weight training movements, based on learning only from the correctly performed routine. We showed why correctly segmenting each repetition of a weight training exercise from a time series data into its repetition signal is important. We designed our workflow based on the signals extracted from a segmentation of the sensor data from correctly performed routines. This lets us use our system in an online environment where the system can detect any anomalies as soon as the end of an exercise repetition has been detected.

Our workflow is based on the concept of the Axis-of-Effect — the direction in which the trainee's main limb moves during an exercise. As a consequence, we highlighted that our algorithm is able to analyse exercises provided the wearable sensor is worn in the appropriate location on the body. Thus the workflow is sensitive to the positioning of the wearable device. We discuss this dependency in more detail in the future work discussion.

Our workflow is based on calculating the prototype representation of an exercise repetition, based on the signals segmented from the time series. Using the derived prototype, the system finds the distribution of the trajectory by mapping each segment to the prototype. Finally, for each newly detected segment of an exercise from the time series, the algorithm checks whether the new trajectory is from the calculated distribution or not. If not the algorithm rejects the segment and alerts the trainee of an anomaly. We show that the workflow can detect whether a given signal that represents a repetition of an exercise has been performed correctly or not. However, a more important question is whether we can identify the incorrect performance of the exercise before the repetition ends. This brings us to the question of how to perform a partial comparison of the repetitions, and the importance of our contribution in computing the average shape of a signal, which we discuss in more detail in the future work section.

In Chapter 4, we formally define weight training activities in terms of bursts of consecutive repeating signals from a time series of measurement data, which we called

an Interval of Recurrence (IoR). To track these IoRs, we designed OToR, a novel and resource-efficient algorithm to find and track intervals of recurrence from time series data streams in real-time on a wearable device. Our algorithm is based on the periodicity and sub-domain linearity properties of autocorrelation, which we formally analyse in Chapter 4 when detecting intervals of recurrence. We provide an in-depth theoretical analysis of these principles. We mathematically prove and experimentally demonstrate the robustness of our algorithm to environmental noise.

We achieve two design objectives for our goal (summarised in Chapter 1 and Table 4.1) that go beyond the state-of-the-art. First, our technique tracks the activity performance with minimal prior knowledge and configuration, i.e., in an unsupervised manner. We achieve this goal by sacrificing the detection of the first three repetitions of an exercise; i.e., OToR waits to receive three repetitions of an exercise before it considers the repetitions to be an exercise set. We can then monitor the exercise after automatically detecting these first three repetitions.

Second, the tracking happens in real-time in a wearable device with low computing and memory resources. We demonstrate the effectiveness of OToR in our experiments in both offline and online scenarios in comparison to several state-of-the-art algorithms. In summary, we have shown that we satisfied the following four design goals:

1. Detecting and tracking weight training activities in real-time,
2. Performing the detection and tracking of exercises in an unsupervised manner,
3. Handling environmental noise, and
4. Adapting to variations based on differences in the performance of exercises by trainees.

In Chapter 5, we further developed our mathematical modelling of weight training activities based on the signals collected from wearable sensors. This chapter advances the idea of comparing repetitions of an exercise based on the shape of their associated

signals, as introduced in Chapter 3. We investigated use of the concept of *Shape* in time series, which enables the comparison of geometric objects by their shape. We introduced the geometric *shape* analogy into time series analysis by describing time series data as a geometric *curve* that is expressed in terms of its relative location and form in space. We then transform the computed curve into a vector space where each signal is presented as points on the surface of a sphere (Shape-Sphere). We prove a pseudo-metric property of shape distances in Shape-Sphere and show how to compute the average shape in this space, for use as a prototype of a set of signals. We demonstrated the effectiveness of these two properties in analysing time series data by applying them in Nearest Centroid Classifiers and K-Means clustering. Our results on 48 publicly available data sets show that Shape-Sphere improves the classification results when shape is the differentiating feature between signals, while keeping the quality of clustering equivalent to current state-of-the-art techniques. We note that the current limitation of Shape-Sphere is that it requires the time series to be of equal lengths before comparing them and we discuss suggested approaches to address this limitation.

In Chapter 6, we demonstrated LiftSmart, a novel smart wearable to detect, track and analyse weight training activities. LiftSmart is the first wearable for weight training monitoring that is based on unsupervised machine learning techniques to eliminate the use of labelled data, to train in a supervised manner. We designed LiftSmart to target users who have knowledge of the weight training activities they are meant to perform, i.e., users of LiftSmart are able to perform their routine correctly. One of the limitations of LiftSmart is that it always evaluates an exercise within its set. The advantages of this unsupervised approach are that it avoids the cost of acquiring labelled training data, and can be trained in a computationally efficient manner. For this to be possible, LiftSmart requires that trainees be able to perform the first three repetitions of an exercise correctly, so that LiftSmart can monitor the rest of the set. In the next section we examine how to address restrictions such as these in future work.

7.2 Future work

In this thesis we have addressed the problem of monitoring weight training activities in real-time, using a wearable device without the use of labelled data. We formally defined the problem and have designed novel machine learning techniques to achieve our goal. To the best of our knowledge we have developed the first wearable device that monitors weight training activities in real-time.

We propose two sets of problems that have been raised by our research, which would be useful for future work: (1) Practical extensions to our application, and (2) Theoretical questions that arise from our theoretical contributions.

7.2.1 Application extensions

In Section 7.1 we discussed the sensitivity of our workflow for detecting unseen anomalies with respect to the Axis-of-Effect. A relevant question for future work is to identify what types of exercises can be addressed by observing data stream from a wearable sensor. That is, if we know the location of a sensor, when should we consider the incoming data as relevant data from the Axis-of-Effect, and when should we consider the data to be irrelevant and hence discard the data? This can be seen as an extension of our method, where multiple sensors are worn by a trainee and we want to identify when data is relevant to the current exercise.

Another extension to our method is that by extending our method to multiple sensors, what information can be extracted from these sensors to help trainees perform their routines correctly? In other words, how can we provide more detailed feedback (other than correct or incorrect performance) to a trainee. For example, during a squat exercise it is ideal to provide detailed feedback to the trainee about the movement of different parts of their body, to provide them with an explanation of why they were unable to maintain their performance.

7.2.2 Theoretical extensions

In Chapter 4 we presented our method OToR to automatically track consecutive repeating signals. Our theory is based on single-dimensional time series. A direct extension of OToR would be to extend the theories to multi-dimension signals. By this extension, we can track consecutive multi-dimensional signals, which in turn can provide greater insight into the quality of repeated activities. For example, in the weight lifting scenario, tracking multi-dimensional signals can lead to sensor fused monitoring systems where trainees wear multiple sensors on different parts of their body to better monitor their overall performance.

In Chapter 5 we presented our method, Shape-Sphere, and showed how to analyse time series data based on their shape. A prerequisite to Shape-Sphere is that all the time series must be of equal length. In Chapter 5 we discussed how the dimensionality of data does not affect its shape. Thus, a possible solution to relax the requirement for equal length signals is to investigate the effect of up-sampling or down-sampling of the data on the Shape-Sphere algorithm.

A second extension to Shape-Sphere is to design methods for partially comparing time series by their shape. In Section 7.1, we noted that our workflow for detecting unseen anomalies can only detect anomalies once the end of a repetition has been seen. In Chapter 5 we discussed how Shape-Series describes the shape of a time series up to a point inside the time series. This behaviour of Shape-Series can be used to design methods that can make a comparison of partial time series, rather than waiting for a repetition to be completed.

Bibliography

[1] "Acc statistics from the 2014/2015 annual report," April 2016. [Online]. Available: http://www.acc.co.nz/about-acc/statistics/

[2] R. T. Olszewski, *Generalized Feature Extraction for Structural Pattern Recognition in Time-series Data*. School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, 2001.

[3] R. Richards, "Final report March 2013," State of Victoria through the Department of Transport, Planning and Local Infrastructure 2013, Technical report 1.1, 2013. [Online]. Available: http://www.sport.vic.gov.au/sites/default/files/Sports_injury_prevention_taskforce_report.pdf

[4] ——, "Cost of sports injuries," Clearinghouse for Sport, Australian Sports Commission, Technical report, 2017. [Online]. Available: https://www.clearinghouseforsport.gov.au/knowledge_base/sport_participation/sport_injuries_and_medical_conditions/sports_injuries

[5] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, January 2017.

[6] D. Morris, T. S. Saponas, A. Guillory, and I. Kelner, "Recofit: Using a wearable sensor to find, recognize, and count repetitive exercises," in *Procceedings of the Conference on Human Factors in Computing Systems (CHI '14)*. New York, NY, USA: ACM, 2014, pp. 3225–3234.

[7] D. Pruthi, A. Jain, K. M. Jatavallabhula, R. Nalwaya, and P. Teja, "Maxxyt: An autonomous wearable device for real-time tracking of a wide range of exercises," in *Procceedings of the 17th International Conference on Modelling and Simulation (UKSim)*, March 2015, pp. 137–141.

[8] C. Shen, B. Ho, and M. Srivastava, "Milift: Efficient smartwatch-based workout tracking using automatic segmentation," *IEEE Transactions on Mobile Computing*, vol. 17, no. 7, pp. 1609–1622, July 2018.

[9] F. Azhar and C. Li, "Hierarchical relaxed partitioning system for activity recognition," *IEEE Transactions on Cybernetics*, vol. 47, no. 3, pp. 784–795, March 2017.

[10] *American Heart Association Recommendations for Physical Activity in Adults and Kids*, American Heart Association, 2018 (accessed October 4, 2019). [Online]. Available: https://www.heart.org/en/healthy-living/fitness/fitness-basics/aha-recs-for-physical-activity-in-adults

[11] *Resistance training – health benefits*, Victoria State Governent, August 2018 (accessed October 4, 2019). [Online]. Available: https://www.betterhealth.vic.gov.au/health/healthyliving/resistance-training-health-benefits

[12] *EU Physical Activity Guidelines: Recommended Policy Actions in Support of Health-Enhancing Physical Activity*, EU Working Group.

[13] M. K. Drew and C. F. Finch, "The relationship between training load and injury, illness and soreness: A systematic and literature review," *Sports Medicine*, vol. 46, no. 6, pp. 861–883, June 2016.

[14] T. J. Gabbett, "The training—injury prevention paradox: should athletes be training smarter and harder?" *British Journal of Sports Medicine*, vol. 50, no. 5, pp. 273–280, 2016.

[15] ——, "Debunking the myths about training load, injury and performance: empirical evidence, hot topics and recommendations for practitioners," *British Journal of Sports Medicine*, vol. 54, no. 1, pp. 58–66, 2020.

[16] T. J. Gabbett, G. P. Nassis, E. Oetter, J. Pretorius, N. Johnston, D. Medina, G. Rodas, T. Myslinski, D. Howells, A. Beard, and A. Ryan, "The athlete monitoring cycle: a practical guide to interpreting and applying training monitoring data," *British Journal of Sports Medicine*, vol. 51, no. 20, p. 1451—1452, October 2017.

[17] "Monitoring athletes taking advantage of technology," Canadian Sport Institute. [Online]. Available: https://www.csipacific.ca/wp-content/uploads/pp/performance-point-TID-1403-monitoring.pdf

[18] E. Velloso, A. Bulling, and H. Gellersen, "Motionma: Motion modelling and analysis by demonstration," in *Proceedings of the Conference on Human Factors in Computing Systems (CHI '13)*. New York, NY, USA: ACM, 2013, pp. 1309–1318.

[19] M. Kellmann, "Preventing overtraining in athletes in high-intensity sports and stress/recovery monitoring," *Scandinavian Journal of Medicine & Science in Sports*, vol. 20, no. s2, pp. 95–102, 2010.

[20] A. W. Salmoni, R. A. Schmidt, and C. B. Walter, "Knowledge of results and motor learning: A review and critical reappraisal," *Psychological Bulletin*, vol. 95, no. 3, pp. 355–386.

[21] M. Isard and A. Blake, "Contour tracking by stochastic propagation of conditional density," in *Proceedings of the European Conference on Computer Vision (ECCV '96)*, 1996, pp. 343–356.

[22] J. Rittscher, A. Blake, and S. Roberts, "Towards the automatic analysis of complex human body motions," *Image and Vision Computing*, vol. 20, no. 12, pp. 905 – 916, 2002.

[23] J. Gall, C. Stoll, E. de Aguiar, C. Theobalt, B. Rosenhahn, and H. Seidel, "Motion capture using joint skeleton tracking and surface estimation," in *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, June 2009, pp. 1746–1753.

[24] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, June 2011, pp. 1297–1304.

[25] I. Ar and Y. S. Akgul, "A computerized recognition system for the home-based physiotherapy exercises using an RGBD camera," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 22, no. 6, pp. 1160–1171, November 2014.

[26] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake, "Efficient human pose estimation from single depth images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2821–2840, December 2013.

[27] K. Buys, C. Cagniart, A. Baksheev, T. D. Laet, J. D. Schutter, and C. Pantofaru, "An adaptable system for RGB-D based human body detection and pose estimation," *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, pp. 39 – 52, 2014.

[28] Y. Chen, C. Shen, X. Wei, L. Liu, and J. Yang, "Adversarial posenet: A structure-aware convolutional network for human pose estimation," in *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 1221–1230.

[29] S. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 4724–4732.

[30] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2019.

[31] R. Khurana, K. Ahuja, Z. Yu, J. Mankoff, C. Harrison, and M. Goel, "Gymcam: Detecting, recognizing and tracking simultaneous exercises in unconstrained scenes," *ACM Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 4, pp. 185:1–185:17, 2018.

[32] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Dense trajectories and motion boundary descriptors for action recognition," *International Journal of Computer Vision*, vol. 103, no. 1, pp. 60–79, May 2013.

[33] F. Anderson, T. Grossman, J. Matejka, and G. Fitzmaurice, "Youmove: Enhancing movement training with an augmented reality mirror," in *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*. New York, NY, USA: ACM, 2013, pp. 311–320.

[34] W. Zhao, H. Feng, R. Lun, D. D. Espy, and M. A. Reinthal, "A kinect-based rehabilitation exercise monitoring and guidance system," in *Proceedings of the 2014 IEEE International Conference on Software Engineering and Service Science*, June 2014, pp. 762–765.

[35] M. Sundholm, J. Cheng, B. Zhou, A. Sethi, and P. Lukowicz, "Smart-mat: Recognizing and counting gym exercises with low-cost resistive pressure sensing matrix," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '14)*. New York, NY, USA: ACM, 2014, pp. 373–382.

[36] J. Cheng, M. Sundholm, B. Zhou, M. Hirsch, and P. Lukowicz, "Smart-surface: Large scale textile pressure sensors arrays for activity recognition," *Pervasive and Mobile Computing*, vol. 30, pp. 97 – 112, 2016.

[37] B. Zhou, M. Sundholm, J. Cheng, H. Cruz, and P. Lukowicz, "Never skip leg day: A novel wearable approach to monitoring gym leg exercises," in *Proceedings of the 2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, March 2016, pp. 1–9.

[38] P. Parzer, A. Sharma, A. Vogl, J. Steimle, A. Olwal, and M. Haller, "Smartsleeve: Real-time sensing of surface and deformation gestures on flexible, interactive textiles, using a hybrid gesture detection pipeline," in *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology (UIST '17)*. New York, NY, USA: ACM, 2017, pp. 565–577.

[39] E. A. H. Akpa, M. Fujiwara, Y. Arakawa, H. Suwa, and K. Yasumoto, "Gift: Glove for indoor fitness tracking system," in *Proceedings of the 2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, March 2018, pp. 52–57.

[40] K.-H. Chang, M. Y. Chen, and J. Canny, "Tracking free-weight exercises," in *Proceedings of the 9th International Conference on Ubiquitous Computing (UbiComp '07)*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 19–37.

[41] H.-T. Cheng, F.-T. Sun, M. Griss, P. Davis, J. Li, and D. You, "Nuactiv: Recognizing unseen new activities using semantic attribute-based learning," in *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '13)*. New York, NY, USA: ACM, 2013, pp. 361–374.

[42] T. T. Um, V. Babakeshizadeh, and D. Kulić, "Exercise motion classification from large-scale wearable sensor data using convolutional neural networks," in *Proceed-*

ings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), September 2017, pp. 2385–2390.

[43] K. Skawinski, F. Montraveta Roca, R. D. Findling, and S. Sigg, "Workout type recognition and repetition counting with cnns from 3d acceleration sensed on the chest," in *Proceedings of the Advances in Computational Intelligence*, 2019, pp. 347–359.

[44] E. Velloso, A. Bulling, and H. Gellersen, "Towards qualitative assessment of weight lifting exercises using body-worn sensors," in *Proceedings of the 13th International Conference on Ubiquitous Computing (UbiComp '11)*. New York, NY, USA: ACM, 2011, pp. 587–588.

[45] J. F. S. Lin and D. Kulić, "Online segmentation of human motion for automated rehabilitation exercise analysis," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 22, no. 1, January 2014.

[46] R. H. Shumway and D. S. Stoffer, *Time series analysis and its applications: with R examples*. Springer, 2017.

[47] D. C. Jonathan and C. Kung-Sik, *Time series analysis with applications in R*, 2008.

[48] H. Deng, G. Runger, E. Tuv, and M. Vladimir, "A time series forest for classification and feature extraction," *Information Sciences*, vol. 239, pp. 142 – 153, 2013.

[49] P. Geurts, "Pattern extraction for time series classification," in *Proceedings of the Principles of Data Mining and Knowledge Discovery*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 115–127.

[50] A. Nanopoulos, R. Alcock, and Y. Manolopoulos, "Information processing and technology." Commack, NY, USA: Nova Science Publishers, Inc., 2001, ch. Feature-based Classification of Time-series Data, pp. 49–61.

[51] F. Dellaert, T. Polzin, and A. Waibel, "Recognizing emotion in speech," in *Proceeding of Fourth International Conference on Spoken Language Processing (ICSLP '96)*, vol. 3, Oct 1996, pp. 1970–1973 vol.3.

[52] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient similarity search in sequence databases," in *Proceedings of the Foundations of Data Organization and Algorithms*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 69–84.

[53] A. Clare and R. D. King, "Knowledge discovery in multi-label phenotype data," in *Proceedinhgs of the Principles of Data Mining and Knowledge Discovery*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 42–53.

[54] Fu-Lai Chung, Tak-Chung Fu, V. Ng, and R. W. P. Luk, "An evolutionary approach to pattern-based time series segmentation," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 5, pp. 471–489, Oct 2004.

[55] V. Megalooikonomou, G. Li, and Q. Wang, "A dimensionality reduction technique for efficient similarity analysis of time series databases," in *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management (CIKM '04)*. New York, NY, USA: ACM, 2004, pp. 160–161.

[56] B.-K. Yi and C. Faloutsos, "Fast time sequence indexing for arbitrary lp norms," in *Proceedings of the 26th International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 385–394.

[57] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Knowledge and Information Systems*, vol. 3, no. 3, pp. 263–286, August 2001.

[58] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proceedings of the 8th Workshop*

on Research Issues in Data Mining and Knowledge Discovery. New York, NY, USA: ACM, 2003, pp. 2–11.

[59] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "Segmenting time series: A survey and novel approach," in *Data Mining in Time Series Databases*, pp. 1–21.

[60] C. Berberidis, I. Vlahavas, W. G. Aref, M. Atallah, and A. K. Elmagarmid, "On the discovery of weak periodicities in large time series," in *Principles of Data Mining and Knowledge Discovery*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 51–61.

[61] J. Paparrizos and L. Gravano, "k-shape: Efficient and accurate clustering of time series," in *Proceedings of the 2015 International Conference on Management of Data*. New York, NY, USA: ACM, 2015, pp. 1855–1870.

[62] S. Papadimitriou, J. Sun, and C. Faloutsos, "Streaming pattern discovery in multiple time-series," in *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB '05)*. VLDB Endowment, 2005, pp. 697–708.

[63] K. Yang and C. Shahabi, "On the stationarity of multivariate time series for correlation-based data analysis," in *Proceedings of the 2005 IEEE International Conference on Data Mining (ICDM'05)*, November 2005, p. 4.

[64] H. Yoon, K. Yang, and C. Shahabi, "Feature subset selection and feature ranking for multivariate time series," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 9, pp. 1186–1198, Sep. 2005.

[65] C. A. Ratanamahatana and E. Keogh, "Making time-series classification more accurate using learned constraints," in *Proceedings of the 2004 SIAM International Conference on Data Mining*, pp. 11–22.

[66] T. Shibuya, T. Harada, and Y. Kuniyoshi, "Causality quantification and its applications: Structuring and modeling of multivariate time series," in *Proceedings of the 2009 International Conference on Knowledge Discovery and Data Mining (KDD '09)*. New York, NY, USA: ACM, 2009, pp. 787–796.

[67] F. K. Chan, A. W. Fu, and C. Yu, "Haar wavelets for efficient similarity search of time-series: with and without time warping," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 3, pp. 686–705, May 2003.

[68] O. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE Communications Surveys Tutorials*, vol. 15, no. 3, March 2013.

[69] T. Fu, "A review on time series data mining," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 164 – 181, 2011.

[70] J. F. S. Lin, M. Karg, and D. Kulić, "Movement primitive segmentation for human motion modeling: A framework for analysis," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 3, June 2016.

[71] P. Dash, M. Nayak, M. Senapati, and I. Lee, "Mining for similarities in time series data using wavelet-based feature vectors and neural networks," *Engineering Applications of Artificial Intelligence*, vol. 20, no. 2, pp. 185 – 201, 2007, special Issue on Applications of Artificial Intelligence in Process Systems Engineering.

[72] V. Jandhyala, S. Fotopoulos, I. MacNeill, and P. Liu, "Inference for single and multiple change-points in time series," *Journal of Time Series Analysis*, vol. 34, no. 4, pp. 423–446.

[73] E. Page, "A test for a change in a parameter occurring at an unknown point," *Biometrika*, vol. 42, no. 3/4, pp. 523–527, 1955.

[74] S. Aminikhanghahi and D. J. Cook, "A survey of methods for time series change point detection," *Knowledge and Information Systems*, vol. 51, no. 2, pp. 339–367, May 2017.

[75] D. Barry and J. A. Hartigan, "A Bayesian analysis for change point problems," *Journal of the American Statistical Association*, vol. 88, no. 421, pp. 309–319, 1993.

[76] S. Chib, "Estimation and comparison of multiple change-point models," *Journal of Econometrics*, vol. 86, no. 2, pp. 221 – 241, 1998.

[77] Z. J. Wang and P. Willett, "Joint segmentation and classification of time series using class-specific features," *IEEE Transactions on Systems, Man, and Cybernetics (Cybernetics)*, vol. 34, no. 2, April 2004.

[78] V. Chandola and R. R. Vatsavai, "A Gaussian process based online change detection algorithm for monitoring periodic time series," in *Proceedings of the 2011 International Conference on Data Mining*, pp. 95–106.

[79] B. Chiu, E. Keogh, and S. Lonardi, "Probabilistic discovery of time series motifs," in *Proceedings of the 2003 ACM International Conference on Knowledge Discovery and Data Mining (KDD '03)*. New York, NY, USA: ACM, 2003, pp. 493–498.

[80] Y. Matsubara, Y. Sakurai, and C. Faloutsos, "Autoplait: Automatic mining of co-evolving time sequences," in *Proceedings of the 2014 International Conference on Management of Data*, 2014.

[81] J. Zhao and L. Itti, "Decomposing time series with application to temporal segmentation," in *Proceedings of IEEE Workshop on Applications of Computer Vision (WACV)*, March 2016.

[82] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: a novel symbolic representation of time series," *Data Mining and Knowledge Discovery*, vol. 15, no. 2, pp. 107–144, Oct 2007.

[83] C. L. Fancourt and J. C. Principe, "Competitive principal component analysis for locally stationary time series," *IEEE Transactions Signal Processing*, vol. 46, no. 11, pp. 3068–3081, 1998.

[84] J. F. Lin, V. Joukov, and D. Kulic, "Human motion segmentation by data point classification," in *Proceedings of the 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Aug 2014, pp. 9–13.

[85] K. Yu, K. Yang, and Y. Bai, "Experimental investigation on the time-varying modal parameters of a trapezoidal plate in temperature-varying environments by subspace tracking-based method," *Journal of Vibration and Control*, vol. 21, no. 16, pp. 3305–3319, 2015.

[86] D. Minnen, T. Starner, I. Essa, and C. Isbell, "Improving activity discovery with automatic neighborhood estimation," in *Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI'07)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007, pp. 2814–2819.

[87] E. Berlin and K. Van Laerhoven, "Detecting leisure activities with dense motif discovery," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing (UbiComp '12)*. New York, NY, USA: ACM, 2012, pp. 250–259.

[88] A. Vahdatpour, N. Amini, and M. Sarrafzadeh, "Toward unsupervised activity discovery using multi-dimensional motif detection in time series," in *Proceedings of the 21st International Jont Conference on Artifical Intelligence (IJCAI'09)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009, pp. 1261–1266.

[89] H. Poincaré, "Sur le problème des trois corps et les équations de la dynamique," *Acta Mathematica*, vol. 13, no. 1, pp. A3–A270, 1890.

[90] J. Eckmann, S. O. Kamphorst, and D. Ruelle, "Recurrence plots of dynamical systems," *World Scientific Series on Nonlinear Science Series A*, vol. 16, pp. 441–446, 1995.

[91] S. Spiegel, J.-B. Jain, and S. Albayrak, "A recurrence plot-based distance measure," in *Proceedings of the Translational Recurrences*, 2014, pp. 1–15.

[92] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh, "Matrix profile I: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets," in *Proceedings of the 2016 IEEE International Conference on Data Mining (ICDM)*, December 2016, pp. 1317–1322.

[93] S. Gharghabi, Y. Ding, C. M. Yeh, K. Kamgar, L. Ulanova, and E. Keogh, "Matrix profile VIII: Domain agnostic online semantic segmentation at superhuman performance levels," in *Proceedings of the 2017 IEEE International Conference on Data Mining (ICDM)*, November 2017, pp. 117–126.

[94] M. Mirmomeni, Y. Kowsar, L. Kulik, and J. Bailey, "An automated matrix profile for mining consecutive repeats in time series," in *Proceedings of the Pacific Rim International Conferences on Artificial Intelligence 2018: Trends in Artificial Intelligence (PRICAI)*. Cham: Springer International Publishing, 2018, pp. 192–200.

[95] R. S. Tsay, *Analysis of financial time series*. John Wiley & Sons, 2005, vol. 543.

[96] X. Zhang, T. Zhang, A. A. Young, and X. Li, "Applications and comparisons of four time series models in epidemiological surveillance data," *PLoS One*, vol. 9, no. 2, 2014.

[97] Q. Li, N.-N. Guo, Z.-Y. Han, Y.-B. Zhang, S.-X. Qi, Y.-G. Xu, Y.-M. Wei, X. Han, and Y.-Y. Liu, "Application of an autoregressive integrated moving average model for predicting the incidence of hemorrhagic fever with renal syndrome," *The American Journal of Tropical Medicine and Hygiene*, vol. 87, no. 2, pp. 364–370, 2012.

[98] I. Batal and M. Hauskrecht, "A supervised time series feature extraction technique using dct and dwt," in *Proceedings of the 2009 International Conference on Machine Learning and Applications*, Dec 2009, pp. 735–739.

[99] F. Korn, H. V. Jagadish, and C. Faloutsos, "Efficiently supporting ad hoc queries in large datasets of time sequences," *ACM Special Interest Group on Management of Data Record*, vol. 26, no. 2, p. 289–300, 1997.

[100] S. Aghabozorgi, A. Seyed Shirkhorshidi, and T. Ying Wah, "Time-series clustering - A decade review," *Journal of Information Systems*, vol. 53, no. C, pp. 16–38, Oct. 2015.

[101] E. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: A survey and empirical demonstration," *Journal of Data Mining and Knowledge Discovery*, vol. 7, no. 4, pp. 349–371, Oct. 2003.

[102] W. H. Abdulla, D. Chow, and G. Sin, "Cross-words reference template for dtw-based speech recognition systems," in *Proceedings of the Conference on Convergent Technologies for Asia-Pacific Region (TENCON 2003)*, vol. 4, 2003.

[103] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani, "Locally adaptive dimensionality reduction for indexing large time series databases," *ACM Transactions on Database Systems*, vol. 27, no. 2, pp. 188–228, 2002.

[104] P. Geurts, "Pattern extraction for time series classification," in *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD '01)*. Berlin, Heidelberg: Springer-Verlag, 2001, pp. 115–127.

[105] M. Gribskov, "Identification of sequence patterns, motifs and domains," in *Reference Module in Life Sciences*. Elsevier, 2018.

[106] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," in *Proceedings of the 1994 International Conference on Management of Data (SIGMOD '94)*. New York, NY, USA: ACM, 1994, pp. 419–429.

[107] K. Buza, J. Koller, and K. Marussy, "Process: Projection-based classification of electroencephalograph signals," in *Proceedings of the Artificial Intelligence and Soft Computing*. Cham: Springer International Publishing, 2015, pp. 91–100.

[108] K.-P. Chan and A. W.-C. Fu, "Efficient time series matching by wavelets," in *Proceedings of the 15th International Conference on Data Engineering*, Mar 1999, pp. 126–133.

[109] Y. Cai and R. Ng, "Indexing spatio-temporal trajectories with Chebyshev polynomials," in *Proceedings of the 2004 ACM International Conference on Management of Data (SIGMOD '04)*. New York, NY, USA: ACM, 2004, pp. 599–610.

[110] H. Ney and S. Ortmanns, "Dynamic programming search for continuous speech recognition," *IEEE Signal Processing Magazine*, vol. 16, no. 5, pp. 64–83, Sep. 1999.

[111] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intelligent Data Analysis*, vol. 11, no. 5, pp. 561–580, Oct. 2007.

[112] Byoung-Kee Yi, H. V. Jagadish, and C. Faloutsos, "Efficient retrieval of similar time sequences under time warping," in *Proceedings of the 14th International Conference on Data Engineering*, February 1998, pp. 201–208.

[113] B. K. Sarker, T. Mori, T. Hirata, and K. Uehara, "Parallel algorithms for mining association rules in time series data," in *Parallel and Distributed Processing and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 273–284.

[114] S. Greco and C. Molinaro, "Approximate probabilistic query answering over inconsistent databases," in *Proceedings of the Conceptual Modeling (ER 2008)*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 311–325.

[115] C. A. Ratanamahatana and E. J. Keogh, "Three myths about dynamic time warping data mining," in *Proceedings of the 2005 International Conference on Data Mining (SDM 2005), Newport Beach, CA, USA, April 21-23, 2005*, 2005, pp. 506–510.

[116] M. Cuturi and M. Blondel, "Soft-dtw: a differentiable loss function for time-series," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 894–903.

[117] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh, "Indexing multidimensional time-series with support for multiple distance measures," in *Proceedings of the 2003 International Conference on Knowledge Discovery and Data Mining (KDD '03)*. New York, NY, USA: ACM, 2003, pp. 216–225.

[118] M.-S. Kim, S.-W. Kim, and M. Shin, "Optimization of subsequence matching under time warping in time-series databases," in *Proceedings of the 2005 ACM Symposium on Applied Computing (SAC '05)*. New York, NY, USA: ACM, 2005, pp. 581–586.

[119] M. Last, Y. Klein, and A. Kandel, "Knowledge discovery in time series databases," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 31, no. 1, pp. 160–169, February 2001.

[120] M. H. Quenouille, "Approximate tests of correlation in time-series 3," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 45, no. 3, p. 483–484, 1949.

[121] B. Podobnik and H. E. Stanley, "Detrended cross-correlation analysis: A new method for analyzing two nonstationary time series," *Physical Review Letters*, vol. 100, no. 8, February 2008.

[122] G. Wachman, R. Khardon, P. Protopapas, and C. R. Alcock, "Kernels for periodic time series arising in astronomy," in *Proceedings of the Machine Learning and Knowledge Discovery in Databases*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 489–505.

[123] L. Gupta, D. L. Molfese, R. Tammana, and P. G. Simos, "Nonlinear alignment and averaging for estimating the evoked potential," *IEEE Transactions on Biomedical Engineering*, vol. 43, no. 4, pp. 348–356, 1996.

[124] F. Petitjean, A. Ketterlin, and P. Gançarski, "A global averaging method for dynamic time warping, with applications to clustering," *Pattern Recognition*, vol. 44, no. 3, pp. 678–693, 2011.

[125] F. Petitjean, G. Forestier, G. I. Webb, A. E. Nicholson, Y. Chen, and E. Keogh, "Faster and more accurate classification of time series by exploiting a novel dynamic time warping averaging algorithm," *Journal of Knowledge and Information Systems*, vol. 47, no. 1, pp. 1–26, 2016.

[126] M. Morel, C. Achard, R. Kulpa, and S. Dubuisson, "Time-series averaging using constrained dynamic time warping with tolerance," *Pattern Recognition*, vol. 74, pp. 77 – 89, 2018.

[127] M. Müller, *Information retrieval for music and motion*. Springer, Berlin Heidelberg, 2007.

[128] I. Pernek, G. Kurillo, G. Stiglic, and R. Bajcsy, "Recognizing the intensity of strength training exercises with wearable sensors," *Journal of Biomedical Informatics*, vol. 58, pp. 145 – 155, 2015.

[129] H. Lieberman and B. Marriott, *Food Components to Enhance Performance: An evaluation of potential performance-enhancing food components for operational rations*. National Academies Press, Chapter 6, 1994.

[130] M. Capecci, M. G. Ceravolo, F. Ferracuti, S. Iarlori, A. Monteriù, L. Romeo, and F. Verdini, "The kimore dataset: Kinematic assessment of movement and clinical scores for remote monitoring of physical rehabilitation," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 27, no. 7, pp. 1436–1448, July 2019.

[131] X. Yun and E. R. Bachmann, "Design, implementation, and experimental results of a quaternion-based Kalman filter for human body motion tracking," *IEEE Transactions on Robotics*, vol. 22, no. 6, pp. 1216–1227, 2006.

[132] R. E. A. Kalman, "New approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[133] J. Jun, R. Guensler, and J. Ogle, "Smoothing methods to minimize impact of global positioning system random error on travel distance, speed, and acceleration profile estimates," *Journal of the Transportation Research Board (Transportation Research Record)*, vol. 1972, pp. 141–150, 2006.

[134] B. M. Yu, K. V. Shenoy, and M. Sahani, "Derivation of extended Kalman filtering and smoothing equations," 2004. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.321.3457

[135] B. J. Mortazavi, M. Pourhomayoun, G. A.-s. Alsheikh, S. I. Lee, and M. Sarrafzadeh, "Determining the single best axis for exercise repetition recognition and counting on smartwatches," in *Proceedings of the IEEE International Conference on Wearable and Implantable Body Sensor Networks'*. IEEE, 2014, pp. 33–38.

[136] E. Velloso, A. Bulling, H. Gellersen, W. Ugulino, and H. Fuks, "Qualitative activity recognition of weight lifting exercises," in *Proceedings of the 4th Augmented Human International Conference (AH '13)*. New York, NY, USA: ACM, 2013, pp. 116–123.

[137] D. Müllner, "Modern hierarchical, agglomerative clustering algorithms," *arXiv preprint arXiv:1109.2378*, 2011.

[138] E. W. Grafarend, *Linear and nonlinear models: fixed effects, random effects, and mixed models*. de Gruyter, 2006.

[139] B. Podobnik and H. E. Stanley, "Detrended cross-correlation analysis: A new method for analyzing two nonstationary time series," *Physical Review Letters*, vol. 100, p. 084102, Feb 2008.

[140] P. H. Colberg and F. Höfling, "Highly accelerated simulations of glassy dynamics using GPUs: Caveats on limited floating-point precision," *Computer Physics Communications*, vol. 182, no. 5, pp. 1120 – 1129, 2011.

[141] D. J. Cook, N. C. Krishnan, and P. Rashidi, "Activity discovery and activity recognition: A new partnership," *IEEE Transactions on Cybernetics*, vol. 43, no. 3, pp. 820–828, June 2013.

[142] D. Patnaik, M. Marwah, R. K. Sharma, and N. Ramakrishnan, "Data mining for modeling chiller systems in data centers," in *Proceedings of the Advances in Intelligent Data Analysis IX*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 125–136.

[143] M. Beyreuther and J. Wassermann, "Continuous earthquake detection and classification using discrete hidden markov models," *Geophysical Journal International*, 2008.

[144] H. Nyquist, "Certain topics in telegraph transmission theory," *Transactions of the American Institute of Electrical Engineers*, 1928.

[145] J.-L. Reyes-Ortiz, L. Oneto, A. Samà, X. Parra, and D. Anguita, "Transition-aware human activity recognition using smartphones," *Neurocomputing*, 2016.

[146] G. Melchiorri and A. Rainoldi, "Muscle fatigue induced by two different resistances: Elastic tubing versus weight machines," *Journal of Electromyography and Kinesiology*, vol. 21, no. 6, 2011.

[147] A. Papoulis, "Error analysis in sampling theory," *Proceedings of the IEEE*, vol. 54, no. 7, pp. 947–955, July 1966.

[148] S. M. Shekatkar, Y. Kotriwar, K. Harikrishnan, and G. Ambika, "Detecting abnormality in heart dynamics from multifractal analysis of ECG signals," *Scientific Reports*, vol. 7, no. 1, p. 15127, 2017.

[149] R. Thuraisingham, "Preprocessing RR interval time series for heart rate variability analysis and estimates of standard deviation of RR intervals," *Computer Methods and Programs in Biomedicine*, vol. 83, no. 1, pp. 78 – 82, 2006.

[150] L. C. M. Vanderlei, C. M. Pastre, R. A. Hoshi, T. D. d. Carvalho, and M. F. d. Godoy, "Basic notions of heart rate variability and its clinical applicability," *Brazilian Journal of Cardiovascular Surgery*, vol. 24, no. 2, pp. 205–217, 2009.

[151] C. E. Kennedy and J. P. Turley, "Time series analysis as input for clinical predictive modeling: Modeling cardiac arrest in a pediatric ICU," *Theoretical Biology and Medical Modelling*, vol. 8, no. 1, p. 40, 2011.

[152] O. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE Communications Surveys Tutorials*, vol. 15, no. 3, pp. 1192–1209, March 2013.

[153] G. Bertoldi, B. A. Burrington, and A. Peet, "Black holes in asymptotically Lifshitz spacetimes with arbitrary critical exponent," *Physical Review*, vol. 80, p. 126003, December 2009.

[154] J. Zhao and L. Itti, "ShapeDTW: Shape dynamic time warping," *Pattern Recognition*, vol. 74, pp. 171–184, 2018.

[155] E. J. Keogh and M. J. Pazzani, "Derivative dynamic time warping," in *Proceedings of the 2001 SIAM International Conference on Data Mining*, 2001.

[156] I. M. Anderson and J. C. Bezdek, "Curvature and tangential deflection of discrete arcs: A theory based on the commutator of scatter matrix pairs and its application to vertex detection in planar shape data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 1, pp. 27–40, January 1984.

[157] M. M. Lipschutz, *Theory and Problems of Differential Geometry*, ser. Schaum's outline series. McGraw-Hill, 1969.

[158] G. Forestier and C. Wemmert, "Semi-supervised learning using multiple clusterings with limited labeled data," *Information Sciences*, vol. 361-362, pp. 48 – 65, 2016.

[159] S.-E. Benkabou, K. Benabdeslem, and B. Canitia, "Unsupervised outlier detection for time series by entropy and dynamic time warping," *Knowledge and Information Systems*, vol. 54, no. 2, pp. 463–486, February 2018.

[160] T. Pelkonen, S. Franklin, J. Teller, P. Cavallaro, Q. Huang, J. Meza, and K. Veeraraghavan, "Gorilla: A fast, scalable, in-memory time series database," *Proceedings of Very Large Data Base Endowment*, vol. 8, no. 12, pp. 1816–1827, 2015.

[161] C. W. Tan, G. I. Webb, and F. Petitjean, "Indexing and classifying gigabytes of time series under time warping," in *Proceedings of the 2017 SIAM International Conference on Data Mining*, pp. 282–290.

[162] G. Forestier, F. Petitjean, H. A. Dau, G. I. Webb, and E. Keogh, "Generating synthetic time series to augment sparse datasets," in *Proceedings of the 2017 IEEE International Conference on Data Mining (ICDM)*, November 2017, pp. 865–870.

[163] L. Kegel, M. Hahmann, and W. Lehner, "Feature-based comparison and generation of time series," in *Proceedings of the 30th International Conference on Scientific and Statistical Database Management (SSDBM '18)*. New York, USA: ACM, 2018.

[164] Y. Kang, S. Chen, X. Wang, and Y. Cao, "Deep convolutional identifier for dynamic modeling and adaptive control of unmanned helicopter," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 2, pp. 524–538, February 2019.

[165] B. J. Jain, "Generalized gradient learning on time series," *Machine Learning*, vol. 100, no. 2, pp. 587–608, Sep 2015.

[166] B. O'Neill, *Elementary differential geometry*. Academic press, 2006.

[167] D. Coeurjolly, S. Miguet, and L. Tougne, "Discrete curvature based on osculating circle estimation," in *Proceedings of the Visual Form 2001*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001.

[168] B. Kerautret, J. O. Lachaud, and B. Naegel, "Comparison of discrete curvature estimators and application to corner detection," in *Proceedings of the Advances in Visual Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 710–719.

[169] D. J. Williams and M. Shah, "A fast algorithm for active contours and curvature estimation," *Computer Vision, Graphics, and Image Processing: Image Understanding*, vol. 55, no. 1, pp. 14 – 26, 1992.

[170] D. Pedoe, *Geometry, a comprehensive course*. Dover Publications, 1988.

[171] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances," *Journal of Data Mining and Knowledge Discovery*, vol. 31, no. 3, pp. 606–660, May 2017.

[172] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, no. 1, pp. 193–218, December 1985.

[173] A. Mueen, E. Keogh, and N. Young, "Logical-shapelets: An expressive primitive for time series classification," in *Proceedings of the 17th ACM International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2011, pp. 1154–1162.

[174] S. Vega-Pons and J. Ruiz-Shulcloper, "A survay of clustering ensemble algorithms," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 25, no. 03, pp. 337–372, 2011.

[175] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana, "Fast time series classification using numerosity reduction," in *Proceedings of the 23rd International Conference on Machine Learning (ICML '06)*. New York, NY, USA: ACM, 2006, pp. 1033–1040.

[176] Z. Geler, V. Kurbalija, M. Radovanović, and M. Ivanović, "Comparison of different weighting schemes for the knn classifier on time-series data," *Knowledge and Information Systems*, vol. 48, no. 2, pp. 331–378, 2016.

[177] T. R. Baechle and R. W. Earle, *Essentials of strength training and conditioning*. Champaign, IL : Human Kinetics, 2016., 2016, ISBN: 9781492501626.