NOVA IMS Information Management School

DOCTORATE PROGRAM

Information Management

Applications of high-frequency telematics for driving behavior analysis

Maria Inês Pastor Pereira da Silva

A thesis submitted in partial fulfillment of the requirements for the degree of Doctor in Information Management

November, 2021

Information Management School
NOVA University Lisbon

Doctoral Program in **Information Management**

Professor Doutor Roberto Henriques, Supervisor

Applications of high-frequency telematics for driving behavior analysis

This document was created using the (pdf)LaTeX processor, based on the "novathesis" template[1], developed at the Dep. Informática of FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

To Fátima and Carlos.

Acknowledgements

This dissertation was only made possible with the help and support of many people. I am grateful for all the guidance and constructive feedback my supervisor, Professor Roberto Henriques, gave me during the Ph.D. Program. Without his help, I would not have completed this work. Our discussions gave me important milestones and helped me keep my work on track.

I also want to thank Professor Fernando Bação, Professor Paulo Rita, and Professor Marco Painho for the insightful seminars they provided. I learned a lot about how to do proper research and how to write and publish articles from them. In addition, I want to highlight how important the feedback and critique I received during these seminars was to me, not only from the teachers but also from my colleagues. Furthermore, I need to mention the comments I got from the anonymous researchers that reviewed my article and provided me with insightful suggestions.

Last but not least, I want to thank my family. My parents were a constant support throughout my life. Their belief in me and their enthusiasm for my success made my life so much easier. My final acknowledgment goes to my partner, Paulo. He was always there throughout this journey, cheering me on, discussing ideas, and staying with me while I worked long hours in random coffee shops. He never complained about the time I had to dedicate to this thesis and was a constant source of motivation and support.

Abstract

Processing driving data and investigating driving behavior has been receiving an increasing interest in the last decades, with applications ranging from car insurance pricing to policy-making. A popular way of analyzing driving behavior is to move the focus to the maneuvers as they give useful information about the driver who is performing them.

Previous research on maneuver detection can be divided into two strategies, namely, 1) using fixed thresholds in inertial measurements to define the start and end of specific maneuvers or 2) using features extracted from rolling windows of sensor data in a supervised learning model to detect maneuvers. While the first strategy is not adaptable and requires fine-tuning, the second needs a dataset with labels (which is time-consuming) and cannot identify maneuvers with different lengths in time.

To tackle these shortcomings, we investigate a new way of identifying maneuvers from vehicle telematics data, through motif detection in time-series. Using a publicly available naturalistic driving dataset (the UAH-DriveSet), we conclude that motif detection algorithms are not only capable of extracting simple maneuvers such as accelerations, brakes, and turns, but also more complex maneuvers, such as lane changes and overtaking maneuvers, thus validating motif discovery as a worthwhile line for future research in driving behavior.

We also propose TripMD, a system that extracts the most relevant driving patterns from sensor recordings (such as acceleration) and provides a visualization that allows for an easy investigation. We test TripMD in the same UAH-DriveSet dataset and show that (1) our system can extract a rich number of driving patterns from a single driver that are meaningful to understand driving behaviors and (2) our system can be used to identify the driving behavior of an unknown driver from a set of drivers whose behavior we know.

Keywords: Driving behaviors, Road safety, Motif Discovery, Acceleration, Sensors, Time-series

Resumo

Nas últimas décadas, o processamento e análise de dados de condução tem recebido um interesse cada vez maior, com aplicações que abrangem a área de seguros de automóveis até a atea de regulação. Tipicamente, a análise de condução compreende a extração e estudo de manobras uma vez que estas contêm informação relevante sobre a performance do condutor.

A investigação prévia sobre este tema pode ser dividida em dois tipos de estratégias, a saber, 1) o uso de valores fixos de aceleração para definir o início e fim de cada manobra ou 2) a utilização de modelos de aprendizagem supervisionada em janelas temporais. Enquanto o primeiro tipo de estratégias é inflexível e requer afinação dos parâmetros, o segundo precisa de dados de condução anotados (o que é moroso) e não é capaz de identificar manobras de diferentes durações.

De forma a mitigar estas lacunas, neste trabalho, aplicamos métodos desenvolvidos na área de investigação de séries temporais de forma a resolver o problema de deteção de manobras. Em particular, exploramos área de deteção de *motifs* em séries temporais e testamos se estes métodos genéricos são bem-sucedidos na deteção de manobras.

Também propomos o TripMD, um sistema que extrai os padrões de condução mais relevantes de um conjuntos de viagens e fornece uma simples visualização. TripMD é testado num conjunto de dados públicos (o UAH-DriveSet), do qual concluímos que (1) o nosso sistema é capaz de extrair padrões de condução/manobras de um único condutor que estão relacionados com o perfil de condução do condutor em questão e (2) o nosso sistema pode ser usado para identificar o perfil de condução de um condutor desconhecido de um conjunto de condutores cujo comportamento nos é conhecido.

Palavras-chave: Perfil de condução, Segurança rodoviária, Descoberta de *motifs*, Aceleração, Sensores, Séries temporais

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ACC Adaptive Cruise Control

ADAS Advanced Driver Assistance Systems

APCA Adaptive Piecewise Constant Approximation

BMU best-matching unit

CA Chebyshev approximation

DFT Discrete Fourier Transformation

DTW Dynamic Time Warping

DTW-SOM Dynamic Time Warping Self-Organizing Map

DWT Discrete Wavelet Transformation

ECG electrocardiography

EMD Extended Motif Discovery

IPLA Indexable Piecewise Linear Approximation

LARFDSSOM Local Adaptive Receptive Field Dimension Selective Self-organizing Map

MDL Minimum Description Length

NDS Naturalistic Driving Studies

OBD On-Board Diagnostics

PAA Piecewise Aggregate Approximation

PAYD pay-as-you-drive

PHYD pay-how-you-drive

SAE Society of Automotive Engineers

SAX Symbolic Aggregate approXimation

SOM Self-Organizing Map

SVD Single Value Decomposition

TWED Time-Warped Edit Distance

UBI Usage-based Insurance

VSAX Variable SAX

Chapter 1

Introduction

1.1 Vehicle telematics

The word *telematics* is a direct translation of the word *télématique*, a concept first introduced by Nora and Minc [1] in their report to the French government about the possible impacts of computerization on society. At the time, telematics referred to the broad connection between telecommunications (*télécommunications*) and computers (*informatique*) [2] and to any services where telecommunications were used to share information. However, nowadays, it is normally associated with *vehicle telematics* and all the technology that enables "sending, receiving and storing vehicle information"[3].

Therefore, sensor generated data is at the core of vehicle telematics. The first collection of sensor data was done with On-Board Diagnostics (OBD) systems. These systems connect to the car's control unit and provide reporting and self-diagnostic features to the car's driver, manufacturer and/or technician. Since the 1980s, OBD systems have suffered many adaptations and they are still a major source of telematics data. Nowadays, they can provided real-time information about various car malfunctions, fuel consumption and emissions, engine rotations, car velocity, and the driver's use of pedals, transmission and steering wheel. The data can then be accessed via USB or Bluetooth.

With the widespread adoption of smartphones and their growing capabilities, the smartphone-base telematics market has been gaining some traction, both in the industry and in the academia [4]. To collect data, the driver simply has to mount his/her smartphone in a fixed place in the car and let the device record and process information from its built-in sensors, which makes it a very cheap implementation. The processed data can be used locally in a smartphone app or sent to a central repository via wireless communication. The sensors used in smartphone-base telematics are usually the GPS and other location services, Accelerometer, Magnetometer, Gyroscope, Microphone and Camera.

In the last two decades, the telematics market has grown significantly and many applications for this technology have been introduced. These include traffic management to improve traffic flow in cities [5, 6], navigation [4], carsharing [7], road condition monitoring [4] and fuel consumption optimization [8]. However, in this work, we'll focus on a specific application of vehicle telematics, namely the analysis of driving behavior. In the next section, the connection between telematics and driving behavior will be further explored.

1.2 Telematics and driving behavior

In the car insurance market, the introduction of telematics created a whole new product line that is currently offered by many insurers.Usage-based Insurance (UBI) is an insurance scheme that relies on telematics to assess the driving of each individual client and to provide discounts accordingly. Instead of using static proxies (such as age,

gender, etc.) filled by customers in questionnaires, insurers can now collect real-time data that expresses the individual behavior of customers and improve their pricing and reserving models.

In both actuarial and transportation research communities, there seems to be a consensus that UBI schemes can have some benefits for insurers and the society in general. In particular, Tselentis et al. [9] identified four benefits of UBI:

- It is fairer to consumers and avoids the use of potentially discriminatory variables, such as gender and age. By using metrics that better describe driving behavior, customers will be priced based on the real risk of having accidents.
- Cross-subsidization is somewhat lessened. In other words, a more personalized pricing reduces the cases where "good drivers"pay higher premiums than they should to compensate the claims generated by "bad drivers".
- Drivers are encouraged to improve their driving performance by reducing high risk behaviors since their premiums would increase in those situations.
- Some schemes provide feedback to drivers, which raises their awareness of high risk behaviors and may ultimately lead to a long-term improvement of driving performance.

With these features, UBI has the potential to make the car insurance market fairer and the transportation system safer. However, these benefits can only materialize if the schemes provide the right incentives to customers. Therefore, the issue of using telematics data to correctly identify risky driving behaviors and model accident risk is extremely relevant in this context.

There are two main types of UBI schemes. The pay-as-you-drive (PAYD) schemes take into account driving habits, such as average mileage and general route, to better access the exposure of that driver, while the pay-how-you-drive (PHYD) schemes use more detailed data, for instance GPS coordinates, velocity and acceleration, to identify and characterize driving behavior.

In practice, because implementing a PAYD is much simpler and requires less complexity, most telematics insurance products currently in the market are based on PAYD schemes [9]. However, driving habits are only one part of the story. Two drivers with similar habits can have completely different risk profiles just because of how they drive. This is the main shortcoming of current PAYD insurance schemes and it is what is pushing experts to have a closer look at PHYD schemes and, as a consequence, strategies that analyze driving behavior.

However, being able to identify driving behavior from telematics is not only relevant for the insurance market. Manufacturers and policymakers can also leverage driving data to understand which factors are associated with accidents and improve road safety with better safety systems and regulation, respectively. This is visible

in the shift research related with accident risk experienced with the introduction of Naturalistic Driving Studies (NDS).

In these studies, participants agree to have their own cars instrumented with small cameras and sensors and, during a specific period of time (usually months), their driving is continuously recorded as they go about their daily routines. Since the aim is to capture natural behaviors, no specific instructions are given to the participants and there are no human observers during the experiment [10].

Note that NDS usually have two main goals, namely collecting data that reflects patterns of *normal* driving behavior, and recording data in the leading moments to a crash. The idea is to either study the prevalence of certain driving behaviors or to find which behaviors are more associated with accidents [11]. Thus, finding better ways of analyzing driving behavior for telematics can also have an impact of road safety and accident risk research.

Finally, this research can also have an impact on two more growing transportation industries. In fleet management, studying the relationship between driving behavior and fuel consumption can improve driving performance and reduce costs. In the self-driving cars industry, analyzing how the cars perform can also help developers understand what is working correctly with the autonomous system and which areas need to be improved.

1.3 Driving behavior and maneuvers

When tackling the driving behavior problem, it is common to start with the maneuvers. This strategy can be successful because the exact maneuvers being performed during a trip and the way they are performed can provide relevant information about the behavior of the driver during the trip. Thus, finding ways of detecting and analyzing maneuvers from telematic data an is important part of designing systems to identify driving behavior. From the papers we could find on this topic, most of them can be aggregated into two main strategies, namely, the fixed thresholds strategy and the rolling windows strategy.

In the fixed thresholds strategy [12, 13, 14, 15], authors set specific thresholds on acceleration or other inertial measurements in order to define the beginning and end of maneuvers. For instance, a left turn could be identified as the interval where the lateral acceleration is higher than 0.3g and the velocity is higher than 30km/h. These types of approaches have the advantage of being easy to implement, lightweight, and interpretable. However, since the choice of the thresholds is essential to the detection of maneuvers, these methods require fine-tuning for each specific dataset and are inflexible to changes in the data.

The second type of strategies [16, 17, 18, 19, 20, 21, 22, 23, 24] involves the use of rolling windows, in which the trip is divided into fixed-sized time windows that can have some level of overlap. Then, each window is used a detection model that

classifies it as a specific maneuver. In this strategy, the window size and the percentage of overlap are important parameters that need to be set beforehand. In a paper testing the rolling window method for identifying simple maneuvers such as braking, turning and acceleration, Xie et al. [19] conclude that the window size is very relevant to the classification performance and that the optimal size can vary depending on the type of maneuver, which leads to the conclusion that an "adaptive window sizing method"would be better suited. Thus, again, the inflexibility of the window size is a disadvantage in this method as well.

Another issue with the rolling windows method is the overlap parameter. If no overlap is used, then the way a trip is split may not be optimal since a window can also split a single maneuver and information may be lost. On the other hand, using the maximal overlap (i.e., moving the window a single time step) comes with its own problems. In their paper, Keogh and Lin [25] argued that clustering time-series subsequences is meaningless when rolling windows are used to build the subsequences. In particular, they state that *"clusters extracted from these time series are forced to obey a certain constraint that is pathologically unlikely to be satisfied by any dataset, and because of this, the clusters extracted by any clustering algorithm are essentially random"*. Thus, one needs to use rule-based detection (which suffers from the same issues as fixed-threshold methods) or one use supervised machine learning models (and this requires labeled data, which is very difficult and time-consuming to collect).

Therefore, with both strategies having their own unique shortcomings, there is a clear space to explore new solutions. Having in mind the need for an adaptable method that can detect maneuvers without the need of labels, we seek to investigate approaches from the time-series data mining community. Firstly, telematics data are, in essence, time-series as they correspond to streams of sensor data. Secondly, the time-series data mining is an established research area with many published works and a large community of researchers.

1.4 Research questions

As previously hinted, the aim of this work is to investigate methods that can detect maneuvers from high-frequency telematics and to understand how the extracted maneuvers relate with driving behavior.

Our hypothesis is that maneuvers are an important component of driving and therefore, extracting maneuvers and analyzing them will allow us to get meaningful insights into the driving behavior of individual drivers.

In particular, we address the following research questions:

- How can we leverage time-series data mining techniques to build an adaptable method that can detect maneuvers from acceleration data without the need of labels?

- How can we summarize the extracted maneuvers in a space-efficient visualization for further analysis?
- Can we use the extracted maneuvers to get insights into the driving behavior of a single driver?

1.5 Contributions

The times-series data mining community is a large area of research and contains numerous tools to analyze time-series data. When researching this field, one tool seemed to fit particularly well the maneuver detection task, namely motif detection. In short, a motif is a subsequence of a time series that repeats itself throughout that same time series. For instance, a single heartbeat in an electrocardiography (ECG) recording would be a motif since the same temporal pattern is repeated throughout the entire ECG time series.

Even though this was not yet fully tested, we hypothesize that a specific maneuver would be a motif in a collection of trips of the same driver. The core idea is that a driver would be consistent in how he performs recurrent maneuvers such as turning or overtaking and, thus, one can detect these maneuvers from the time series of acceleration recordings using motif detection approaches.

Before going into detail about the direction taken, it is important to note that throughout this work, we use a naturalistic driving dataset published by a group of researchers from Spain. It is called the UAH-DriveSet [26] and it contains trip recordings from six different drivers in two specific routes in Madrid, Spain. The authors asked each driver to repeat two predefined routes simulating three different behaviors, namely, normal, aggressive and drowsy. During the trips, they collected both raw and processed signals using the DriveSafe app [27, 28]. The dataset also contains video recordings taken from the smartphone, which can be used to validate results.

In our path to explore motif detection approaches to tackle the problem of maneuver extraction from telematics data, we start by investigating whether motif detection is indeed capable of extracting meaningful maneuvers from acceleration data (Chapter 3). In particular, we implement a modified version of a classical motif detection algorithm, apply it to the UAH-DriveSet and perform a systematic exploration of the resulting motifs. Here we conclude that the motif detection algorithm is capable of extracting simple maneuvers such as accelerations, brakes, and curves, but also more complex maneuvers, such as lane changes and overtaking maneuvers. In turn, these results validate our claim that motif discovery can indeed be used to extract maneuvers.

In this initial work, we also note that the number of motifs extracted from the UAH-DriveSet trips is too large to explore manually. Thus, in Chapter 4, we propose a

summarization algorithm, DTW-SOM, to cluster the motifs extracted by any desired motif discovery algorithm. DTW-SOM is an adaptation the SOM algorithm, a well-known feature reduction and visualization algorithm). In order to work for time-series subsequences, with possibly variable lengths and multiple dimensions, the original SOM algorithm was changed to use the DTW distance as its similarity metric and two specific initialization routines for the SOM network were proposed. We apply DTW-SOM to three different datasets (not related to telematic data) and conclude that our method is capable of extracting relevant information from a set of motifs and display it in a visualization that is space-efficient.

Finally, in Chapter 5, we put the work done in the previous two chapters together and propose TripMD, a system that extracts the most relevant driving patterns from sensor recordings (such as acceleration) and provides a visualization that allows for an easy investigation. We test our system using the UAH-DriveSet dataset and show that it can extract a rich number of driving patterns from a single driver that are meaningful to understand driving behaviors. Interestingly, we also note that TripMD can be used to identify the driving behavior of an unknown driver from a set of drivers whose behavior we know.

Chapter 2

Literature Review

2.1 Driving behavior and styles

2.1.1 Human driving

What exactly is driving behavior? Ever since cars became ubiquitous, driving and road safety have been widely researched and, in the last decades, many advances have been made towards a clear understanding of the factors and behaviors that explain road accidents. Even though there is a common belief that the actions a driver takes have an impact on his accident risk, literature is still fragmented between different fields (e.g. transportation and actuarial sciences).

Therefore, it is very difficult to find generally accepted concepts and frameworks across the literature. After an in-depth review of the literature related with driving behaviors and road safety, Sagberg et al. [29] tried to address this issue by proposing a high-level framework aimed at conceptualizing and understanding what they called *driving styles*. In particular, they defined driving style as "a habitual way of driving, which is characteristic for a driver or a group of drivers".

Although seeming somewhat straightforward, this definition raises an interesting point. The term *habitual way of driving* suggests that a driving style is a set of behaviors a driver shows *consistently* across varied situations and thus we can look at the issue of driving styles from two perspectives. On one hand, it is important to rate driving styles according to accident risk since we are identifying which groups of drivers are consistently riskier. On the other hand, it could be also interesting to recognize when a driver with a low-risk driving style presents irregular behaviors that are outside of his driving style as it could lead to an increased risk in respect to the low-risk driving style.

Sagberg et al. [29] further synthesized previous research by categorizing driving styles and presenting examples of indicators and measures that were used in the literature. They began by differentiating between specific and global driving styles. The first comprehends single behaviors, such as speeding or jerky driving, and can be measured by few indicators. Alternately, global driving styles correspond to general styles manifested by more than one specific driving style and which can only be understood by a variety of indicators. As an example, the most common global driving style is *aggressive driving*, which can be perceived by specific behaviors such as speeding, tailgating and excessive honking.

In relation to specific driving behaviors, which tend to be easier to define, Sagberg et al. [29] divided them into six categories, namely, longitudinal control, lateral control, gap acceptance, errors and violations, visual behavior and, finally, other. Table 2.1 presents the most common examples of specific driving styles for each category, provides a summary of potential motives behind each style (as presented by Sagberg et al. [29]) and summarizes the main conclusions found in the literature relating those styles and accident risk.

Table 2.1: Examples of specific driving styles, underlying motives and relations with accident risk

Category	Specific driving styles	Possible motives	Relation with accident risk
Longitudinal control	Speeding and hash acceleration	Expediency, group pressure, sensation seeking and inattention	Good base of research confirming that, in general, high speeds are associated with high accident risk. Alternately, Waard et al. [30] concluded that, when merging at lower speed, elderly drivers would experience higher risk.
	Jerky driving	Expediency, aggression and inattention	Based on jerk, Desai and Haque [31] built the "spikiness index" to quantify alertness. Murphey et al. [18] used information from the jerk profile to classify drivers.
	Tailgating	Expediency and aggression	Sagberg et al. [29] reported a finding by Bukasa and Risser (1985) where "too short distance to car ahead" was positively correlated with recorded accidents. According to MacAdam et al. [32], tailgating can be also used as an indicator of aggressiveness.
Lateral control	Variable lateral position	Expediency and inattention	This style is mainly used to complement speeding behaviors.
	Speeding in curves	Expediency, group pressure, sensation seeking and inattention	This style is mainly used to complement speeding behaviors. Additionally, it is a theoretically good indicator of accident risk, mainly for low-friction roads.
Gap acceptance	Short gaps in crossings or over-takings	Expediency	Literature focuses on this style's relation with age and aggressiveness.
Visual behavior	Narrow eye fixating	Inexperience and impairment	Literature focuses on this style's relation with age and experience
	Engagement in secondary tasks	Group pressure	Dingus et al. [33] reported that the risk of engaging in distracting activities was two times higher than the baseline.
Errors and violations	Violating traffic lights, stop signs, etc.	Expediency, group pressure and inattention	Based on policy reports, Junger et al. [34] reported a strong association between accident risk and involvement in traffic crime.
Other	Inappropriate honking	Aggression	Literature focuses on this style's relation with aggressiveness

Until now, driving styles related with speeding are the most researched, which is not surprising. In fact, many papers point a clear relationship between this style and accident risk, with high speed profiles usually associated with high accident risk. Nevertheless, as Sagberg et al. [29] pointed out, speed alone cannot explain all aspects of accident risk and the relationships between other driving styles and accident risk need to be further researched.

Early works by Robertson et al. [35] suggested that longitudinal and lateral acceleration could be used to build an "acceleration signature"aimed at characterizing driving styles and identifying drivers. The main hypothesis here was that safe and skilled

drivers are not prone to show extreme acceleration, in other words, high acceleration signatures could detect both sensation seeking and unskilled drivers.

On a similar trend, Wåhlberg [36] built on past research by proposing the concept of *driving celeration behavior* as an overall predictor of accident risk. The author defined it as "the sum or average of all (absolute) accelerations and decelerations (changes in speed) made by a vehicle, in any direction on a two-dimensional plane"and suggested that all behaviors which result in speed variations (namely speed, close following, braking and steering control actions) contribute to the overall driving celeration.

Both approaches are quite interesting because instead of measuring specific driving styles separately, the authors propose general indicators which incorporate many styles. It is interesting to note that most categories proposed by Sagberg et al. [29] can be detected using acceleration and velocity data. This means that a simple high-frequency telematics device that only collects these data can already provide meaningful insights into driving behavior.

Recent studies have looked more closely at inattentive behaviors. According to Dingus et al. [33], crash causation has experienced a shift, with distraction being a fundamental factor for road safety. On a similar note, Dozza [37] concluded that some behaviors associated distraction, namely attendance to secondary tasks and eyes-off-road, reduced significantly drivers' response time in near-crash settings.

2.1.2 Autonomous systems

Another exciting area in driving behavior research is the analysis of how self-driving cars operate (both in real and simulated conditions) and how human drivers interact with automated driving systems.

Even though the idea of autonomous cars has existed for a long time, the switch from a fully human-operated road to a fully machine-operated road is a gradual process. One of the most used taxonomies for describing this gradual change is one published by the Society of Automotive Engineers (SAE). In their report [38], the SAE sets six levels of driving automation, namely:

- Level 0 - No Driving Automation
- Level 1 - Driver Assistance
- Level 2 - Partial Driving Automation
- Level 3 - Conditional Driving Automation
- Level 4 - High Driving Automation
- Level 5 - Full Driving Automation

In level 0, cars are controlled at all times by the human driver. Although there may be some auxiliary systems, such as an emergency braking system, the driving is always done by the driver since these systems do not technically perform driving functions.

In levels 1 and 2, we start seeing some automation, however, the driver is still in charge of monitoring the road and making most decisions. A classical level 1 system is the Adaptive Cruise Control (ACC), which was introduced in the 1990s. These systems are designed to maintain road safety by automatically keeping a safe distance from the car in front and ensuring that the car travels within the speed limits. In other words, the system would change the car's speed without the driver's intervention.

On the other hand, a level 2 system is capable of performing multiple automation tasks, such as controlling both the steering and the speed of the car. Here the automation falls short of self-driving since the driver is still in the seat and must take control of the car in response to changes to the environment. A common example of level 2 are the Advanced Driver Assistance Systems (ADAS).

ADAS were introduced to address some of the road accidents caused by human error. Even though not all of the systems include the full list, ADAS include a wide range of safety applications, including, pedestrian detection and avoidance, automatic emergency braking, blind spot detection, traffic sign recognition, and lane departure warning/correction.

With driver inattention being an ever-increasing source of road accidents (as we discussed in the previous subsection), these systems had the potential of improving road safety by automating some important safety mechanisms. However, previous work investigating how human drivers interact with level 1 and 2 systems leads to the conclusion that more automation in cars may cause a reduction in drivers' attention, which in turn contributes to an impaired driver performance during system failures [39, 40].

Interestingly, this finding highlights that driving automation does lead to unintended (and sometimes unexpected) effects. And this is one way that driving behavior analysis can help with the shift to a road with fully autonomous cars. In the cases where humans still need to make driving decisions and interact with autonomous systems, understanding driving behavior will help manufacturers and policy makers improve the systems to avoid the pitfalls of driver inattention.

What about vehicles with higher levels of automation that do not require as much driver intervention? From level 3 onwards, the automation systems start to make some of the driving decisions based on the environment they are experiencing. However, in level 3, the system cannot intervene in unexpected circumstances or the case of system failures. Thus, in level 3, even though the car will drive most of the time, the driver still needs to be in the seat and maintain awareness of the environment to act in unexpected situations.

Using a simulation environment, Jamson et al. [41] analyzed how drivers responded in a highly automated system versus a manual system. They concluded that

participants experiencing high automation were less prone to retake control of the car for certain tasks such as overtaking. They also observed that safety margins in car following were improved in light traffic conditions. However, in the scenario of high automation, drivers were more distracted by in-vehicle entertainment tasks than the drivers in manual driving, which led to a decreased attention to the road. On a similar note, Merat et al. [42] states that level 3 automation systems do lead to a decrease in drivers' attention and slower response times to retake control of the car, which could be an issue in the case of a system failure.

Finally, in levels 4 and 5, the system is capable of intervening if anything unexpected happens or if there is a system failure. The difference between the two is that in level 4 the drive still has the option of overriding the system's decisions, while in level 5 humans are merely passengers and cannot overriding decisions.

At these levels, understanding the driving behavior of a human driver is not as important since the car will do all the driving. However, analyzing the driving performance of these systems will still be relevant as a means of diagnosing problems and improving them.

2.1.3 Maneuver detection

In the previous two subsections, we looked into driving behavior in general terms, either in the case when a human driver is fully in control or when there is some sort of automation. However, we have not yet discussed how to identify driving behavior from telematics data.

From the available research, many authors have proposed solutions for the task of identifying maneuvers from high-frequency trip recordings. In general terms, all methods can be split into two main strategies. Table 2.2 contains a summary of previous work on this topic and maps each article to the respective strategy.

The first strategy, which we refer to as fixed thresholds, involves the use of predefined thresholds set on the acceleration and/or other inertial measurements to define when each maneuver begins and ends. For instance, Paefgen et al. [13] defined thresholds on the lateral acceleration to identify left and right turns, and used threshold on the longitudinal acceleration to detect acceleration and braking maneuvers. On a slightly different approach, Johnson and Trivedi [12] computed a simple moving average of the rotational energy of smartphone sensors and utilized thresholds on this metric to identify a general maneuver.

The fixed thresholds strategy has the benefit of being easy to implement, easy to interpret, and fast. However, since the choice of the thresholds is fundamental to the detection, these methods require fine-tuning for each specific dataset and are inflexible to changes in the data.

In the second strategy, the authors use rolling windows to split the trip recordings. More specifically, each trip is divided into fixed-sized time sequences that can

Table 2.2: Overview of maneuver detection methods

Strategy	Method specifics	Reference
Fixed thresholds	Use of rotational energy to detect different maneuvers	[12]
	Use of lateral acceleration to detect turns and longitudinal acceleration to detect braking and acceleration	[13]
	Use of rotational energy to detect different maneuver	[15]
	Combination of acceleration and other signals to detect different maneuvers	[14]
Rolling windows	Window sizes of 15, 9, 6 and 3 seconds	[18]
	Window sizes from 0.5 to 5 seconds	[21]
	Window sizes of 20, 50 and 100 time steps	[22]
	Window sizes of 0.2 seconds	[23]
	Small size (5-30 seconds) and medium size (1-30min) windows with no overlap	[17]
	Window sizes from 2 to 7 seconds	[24]
	Windows with 64 time steps and with 50% overlap	[16]
	Windows with 7 time steps	[20]
	Window sizes from 0.5 to 5 seconds, with no overlap	[19]

have some level of overlap, and then each sequence is fed into a supervised learning algorithm that predicts which maneuver is being performed if any. It is important to note that some authors do the prediction directly on the rolling windows while other introduce feature extraction step before the prediction.

Xie et al. [19] test different variations of this second strategy. Particularly, the authors evaluate various feature extraction methods in the task of detecting maneuvers such as brakes, turns, and accelerations. In the definition of the rolling windows, they used several window sizes, but always without overlap. The conclusion there was that the window size is very important for the overall detection performance and that different maneuvers are better detected using different window sizes. In addition, they suggest the use of an "adaptive window sizing method"to address this. This work highlights one of the main disadvantages of the rolling windows strategy, namely, the inflexibility of the window size.

The degree of window overlap is another relevant parameter that authors treat differently. For instance, Saleh et al. [16] built windows from inertial sensors and classified them as aggressive, normal, or drowsy using a stacked-LSTM model. Here they used windows of 64 time-steps and a 50% overlap. In contrast, Weidner et al. [17] used windows of acceleration and velocity with no overlap, computed features from those windows, and fed them to a classifier.

The reason why there are very different approaches to the overlap parameter in the literature is that there is no obvious setting that works best. If no overlap is used, there is no guarantee that each maneuver will be contained a single window. In other words, the windows will likely split some maneuvers and information will be lost as a consequence. However, increasing the level of overlap comes with its issues.

To this point, Keogh and Lin [25] argue that clustering time-series subsequences built from rolling windows with overlap is essentially meaningless. Specifically, they state that *"clusters extracted from these time series are forced to obey a certain constraint that is pathologically unlikely to be satisfied by any dataset, and because of this, the clusters extracted by any clustering algorithm are essentially random"*.

It should be noted that this problem only appears when using clustering (i.e., an unsupervised learning algorithm). This means the one needs to use a supervised approach when using rolling windows with high levels of overlap. However, labeling maneuvers in high-frequency telematics datasets is a massive job and thus the datasets available are extremely scarce. So, and because using unsupervised or semi-supervised methods is quite helpful, for this specific problem, rolling windows does not seem to be an optimal strategy.

The paper from Keogh and Lin [25] goes one step further by also proposing a possible solution to the problem of clustering rolling windows, namely, using time-series motifs as the input for the clustering algorithm. This idea triggered us to look more closely into the field of time-series data mining as an avenue of approaches for maneuver detection, and, in particular, to look more closely at the area of motif detection.

However, before defining more concretely what time-series motifs are and how they can be extracted, we need to start with two major areas in time-series data mining, i.e, representation methods and similarity measures. Thus, in the next section, we will discuss these two topics and, afterward, delve deeper into motif detection.

2.2 Time-series Data Mining

It is clear that telematics data can be very useful when studying driving behavior and accident risk. However, high-frequency sensor data comes with its own issues. Firstly, sensor data is very prone to noise, which can distort results. Secondly, due to its high dimensionality, working with raw sensor data can be expensive, both in terms of computation time and storage. Thirdly, measuring similarity between two streams of sensor data can be very challenging since the time structure of each stream is extremely relevant.

However, these issues are not exclusive of high-frequency sensor data as most time series analysis, in general, encounter these same problems [43]. Given the pervasiveness of time series and their potential for applications, this field of research is somewhat mature and many advancements have been made in the last few years. For the sake of conciseness, three main topics related with time-series data mining will be reviewed, namely representation methods, similarity measures and time-series motifs.

2.2.1 Representation methods

Representation methods correspond to transformations that reduce the dimensionality of a dataset without losing relevant information or characteristics of that dataset. Since most time-series data mining tasks are computationally insensitive, using representation methods to reduce dimensionality is the first step in many tasks in this area. Besides dimensionality reduction, some methods can be also useful for noise reduction.

Wang et al. [43] implemented eight different representation methods, which are summarized in table 2.3, and compared their indexing effectiveness in 38 publicly available time series data sets. The authors tentatively concluded that, even though there was not a clearly superior method, some classes of representations were more effective for specific time series. For example, Wang et al. [43] stated that "on a highly periodic data set the spectral methods are better, whereas on bursty data sets Adaptive Piecewise Constant Approximation (APCA) can be significantly better."

Additionally, Symbolic Aggregate approXimation (SAX) [44] is the one of the most widely used representation method in practice. It has three main advantages, (1) it is simple, (2) requires very little memory and (3) provides indexing with a lower-bounding distance measure. SAX transforms sliding windows of the original time-series into discrete sequences of characters. Since each sliding window is encoded with a predefined number of characters, the continuous time-series is broken into a list of fixed-sized sequences of characters (or words).

Table 2.3: Summary of main representation methods for time series

Method	Acronym	Reference	Description
Discrete Fourier Transformation	DFT	[45]	It converts time series into their discrete frequency representation. This method approximates the series with sinusoids.
Chebyshev approximation	CA	[46]	It is a polynomial approximation using as the orthogonal basis Chebyshev polynomials. This approximation is very close to the minimax polynomial, the theoretical approximating polynomial that has the smallest maximum deviation from the function being approximated.
Haar's Discrete Wavelet Transformation	DWT	[47]	This transformation is very similar to Fourier-based transformations. However, instead of using trigonometric functions to approximate the original time series, it uses Haar's wavelet, which is a sequence of squared-shaped functions that form an orthonormal basis.
Piecewise Aggregate Approximation	PAA	[48]	It is an approximation where equal-length segments of the time series are represented by the mean of the points falling within the segment.
Adaptive Piecewise Constant Approximation	APCA	[49]	In this method, time sequences are approximated by constant segments in such a way that minimizes the reconstruction error.
Single Value Decomposition	SVD	[50]	It applies a linear transformation to the original time series (rotation and scaling) in such a way as to maximize the discriminatory power of the first components. It is a generalization of Principal Component Decomposition.
Symbolic Aggregate approXimation	SAX	[44]	It is a symbolic representation of SAX, where each segment is represented by a character based on the level of its mean value.
Indexable Piecewise Linear Approximation	IPLA	[51]	This method corresponds to a piecewise linear approximation of the time sequences with a new distance function that permits indexing.

2.2.2 Similarity measures

The analysis on similarity measures was based on two comparative studies where the authors tested the accuracy of some similarity measures on a big set of publicly available time series datasets. They both applied a one-nearest neighbor classifier using the similarity measures under test and computed the accuracy of the resulting model. This is a standard choice as the error of this classifier depends heavily on the similarity measure used. Table 2.4 shows the main measures tested by Wang et al. [43] and by Serrà and Arcos [52].

There are two main families of similarity measures for time series, namely, lock-step and elastic. As the name suggests, lock-step measures make a one-to-one comparison of single points from the same time location. The most common examples are the L_p-norms, such as the Euclidean distance and the Manhattan distance. They

Table 2.4: Summary of similarly measures tested by Wang et al. [43] and Serrà and Arcos [52]

Type	Measure	Reference	Wang et al.	Serrà & Arcos
Lock-step	Euclidean distance	[53]	X	X
	DISSIM	[54]	X	
Elastic	Dynamic Time Warping	[55]	X	X
	Longest Common Subsequence	[56]	X	
	Edit Distance with Real Penalty	[57]	X	
	Edit Distance on Real sequence	[58]	X	X
	Sequence Weighted Alignment model	[59]	X	
	Time-warped edit distance	[60]		X
	Minimum jump costs dissimilarity	[61]		X
Other	Spatial Assembling Distance	[62]	X	
	Search based on Threshold Queries	[63]	X	

are quite used because they are easy to understand, are parameter-free and have a linear complexity. However, due to their inflexibility, these measures cannot adapt to time series with different lengths or frequencies (and thus one time series needs to be re-sampled to the size of the other) and are blind to situations where the time series are misaligned in time.

From their analysis, Wang et al. [43] concluded that even though it could be sensitive to noise and time-shifts, the Euclidean distance compared surprisingly well with other more complex measures. In fact, for large datasets, the accuracy and speed of this measure seemed to converge to the ones showed by the best performing measures. Conversely, Serrà and Arcos [52] reported that in general the Euclidean distance performed statistically worse than other elastic measures. It should be noted that time misalignments are frequent in driving data and thus this specific case may have just enough noise to render lock-step measures insufficient.

On the other hand, elastic measures aim at mitigating this drawback by allowing a more flexible mapping between points in time. The first example is the DTW measure, which was briefly presented in the previous section. The DTW was introduced in the time series research community by Berndt and Clifford [55] and since then different authors proposed ways of improving its accuracy and efficiency, making the DTW a rather competitive option. This was supported by Wang et al. [43], who concluded that, on average and across varied datasets, the DTW was no worse than other more recent measures.

Another well known group of elastic measures are the editing measures. They were developed as an extension of the original edit distance [64], which was introduced to compute dissimilarity in strings. There are a quite a few of these measures, although from the two comparative studies mentioned before, they seem to be more or less comparable in terms of accuracy. According to Serrà and Arcos [52], there was an

exception. The Time-Warped Edit Distance (TWED) from Marteau [60] proved to consistently outperform all other measures. The TWED has the advantage of being a metric and thus being capable of exploiting the triangular inequality for speeding up searches. Additionally, Marteau [60] introduced a stiffness parameter which controls the degree of the time shifting allowed.

Finally, it should be noted that both comparative studies conceded that there were always cases where a general low-performing measure could have the best performance in a specific dataset.

2.2.3 Time-series motifs

The concept of motifs in time-series was first introduced by Lin et al. [65] and is usually associated with over-represented segments in a time-series. More precisely, a motif is group of time-series segments, or subsequences, that meets three conditions [66]:

- **Behavior constrain** - This condition states that subsequences should only belong to the same motif if they present the same general behavior. According to [67], distortions such as noise or time and amplitude shifts can be accepted, depending on the application.
- **Distance constrain** - This condition states that the subsequences of a motif must have a distance smaller than a predefined radius R to the center of the motif, where the center is defined as the subsequence that best represents the motif. The radius R is one of the most important parameters any motif detection algorithm and it usually needs to be defined by the user through expert judgment.
- **Non-overlapping constrain** - This condition states that two subsequences in the same motif cannot overlap in time. The condition aims to avoid trivial matchings [65], which are subsequences that share most of the same observations.

From this general definition, several authors tackled the problem of detecting motifs in time-series from diverse perspectives. More precisely, there are three dimensions in which the task of motif detection can vary from author to author, namely the definition of motif relevance, the type of search, and the subsequence length type. In the next paragraphs, we'll go into more detail about these problem dimensions.

The motif relevance states how important a motif is. In other words, for all groups of time-series subsequences that meet all the conditions to form a motif, how can we find which groups are more important? According to Mueen [68], there are two main ways of defining the relevance of a motif:

- **Support-based** - the most relevant motifs are the ones with the largest number of subsequences.

- **Similarity-based** - the most relevant motifs are the ones where the subsequences have the highest similarity between them (i.e., the lowest distance).

Thus, while similarity-based definition results in highly similar motifs, the support-based definition results in highly frequent motifs.

There are two types of searches in the motif detection problem. The first corresponds to discovering the most relevant motif, which in turn depends on the definition of relevancy discussed int the previous article. In the second type, instead of focusing on the single most important motif, the search aims at extracting the k most important motifs or, as usually referred to in the literature, the *k-motifs* [65]. In this search type, one orders all the groups of subsequences that meet the requirements of a motif and takes the first k motifs. However, in order to avoid extracting motifs with overlapping subsequences, there's an additional distance constrain in this search type. For any two motifs in the *k-motifs*, their centers must have a distance higher than $2R$, where R is the radius that sets which subsequences are considered to belong to each motif.

The final dimension is related to the subsequence length type [67]. In other words, how do you define the size of the subsequences that make each motif? There are two possible length types - fixed or variable. The first motif detection algorithms started to address the fixed-length problem, where the goal was to find motifs with a predefined length. Therefore, the algorithms assumed that motifs had a fixed size that was known beforehand. However, for some applications, assuming all motifs have the same size can be too restrictive and, thus, one needs to look for variable-length motifs. In this case, we don't know beforehand the exact size of all motifs and, instead, we search for motifs in a range of motif sizes. This added flexibility comes with a cost as the search space is larger and more complex.

In every flavor of motif detection, there is a prevalent problem. Because motif detection requires the comparison of all possible pairs of subsequences from a time-series, it is a computationally demanding task. Consequently, all motif detection algorithms need to find a strategy to speed up the search.

One of the most popular solutions is to use a low dimensional representation of the original time-series where the distance of subsequences in the original representation is similar [68]. With this dimensionality reduction, one can extract motif candidates in the low-dimensional representation (which is more efficient than using the original representation) and filter the candidates based on the real distance in the original representation. Common representation methods used in motif detection algorithms include SAX [44] and Discrete Fourier Transformation (DFT) [45], methods that were discussed in Subsection 2.2.1.

Another widely used approach to speed-up the search is the Matrix Profile (or MP) [69]. The MP is a meta time-series that annotates the original time-series by providing the distance and the index of each fixed-size sequence's nearest neighbors, excluding

trivial matches. There are a few efficient implementations of the MP, either approximate or exact, and, once the MP is computed, extracting similarity-based motifs is trivial.

In addition to the speed-up strategy, the other essential component of any motif detection algorithm is the choice of the distance function. In other words, how does the algorithm measures the distance between two subsequences? Most authors use the Euclidean distance [53] or the DTW distance [55], which were discussed in subsection 2.2.2.

Finally, motif detection algorithms can be further divided into two groups, depending on the dimensionality of the time-series. The simplest case is to find motifs in one-dimensional time-series. However, for some use-cases, the time-series contain more dimensions, which requires algorithms that can find multidimensional motifs.

To solve the multi-dimensionality problem, different authors proposed varied approaches, ranging from using dimensionality reduction techniques coupled with a one-dimensional motif detection algorithm [66] to applying a motif detection algorithm to every dimension independently and looking for co-occurrences to extract the multidimensional motifs [70, 71, 72, 73].

Chapter 3

Article no. 1 - Finding maneuver motifs in vehicle telematics

This chapter contains the first article of the thesis. This article was published in the journal *Accident Analysis & Prevention* and can be accessed at https://doi.org/10.1016/j.aap.2020.105467.

The main contribution of the article is to validate our hypothesis that motif detection cab be used to detect maneuvers and other driving patterns from high-frequency telematic data. In particular,

- We review the main algorithms to detect time-series motifs and do a critical analysis on their feasibility for detecting maneuvers.
- We make our own implementation of one of the motif detection algorithms.
- We explore the relationship between the motifs from the UAH-DriveSet dataset and the respective maneuvers being performed.

3.1 Introduction

Driving behavior has a great impact on road safety. According to Dingus et al. [33], accident causation has experienced a shift, with driver-related factors being a fundamental factor for road safety. On a similar study, Dozza [37] found that some behaviors associated with distraction, such as attendance to secondary tasks and eyes-on-road, reduce significantly drivers' response time in near-accident settings and, therefore, distraction and inattention are a major factor in traffic accidents.

On the other hand, because data acquisition systems such as smartphones or OBD devices are easier to access and data is cheaper to store and process, researchers have the opportunity of using vehicle telematics to better understand driving behaviors and the factors contributing to car accidents [11].

Driving behavior analysis also has impact for practitioners. In the insurance market, driving behavior has been used through the implementation of UBI [9]. These schemes rely on collecting continuous streams of cars' sensor data on single customers and using that data to assess their driving behavior and provide discounts on the premiums. Other applications include fuel consumption optimization, fleet management and evaluation of self-driving cars' performance.

When tackling the driving behavior problem, it is common to focus on maneuvers. The exact maneuvers being performed during a trip and the way they are being performed is very informative of the driving behavior of the driver. In this context, the question of how to use telematics to correctly identify and compare maneuvers needs to be addressed. Previous research has dealt this issue through two main strategies, namely, 1) using fixed thresholds in inertial measurements (e.g. acceleration or rotational energy) to define the start and end of specific maneuvers or 2) using features extracted from rolling windows of sensor data (e.g. velocity and acceleration) in a supervised learning model to detect maneuvers. While the first method cannot adapt to small fluctuations in the signal and requires fine-tuning, the second requires a dataset with labels indicating where the maneuvers appear and cannot identify maneuvers with different lengths in time.

Having in mind the need for an adaptable method that can detect maneuvers without the need of labels, we sought to investigate a type of methods created in the time-series data mining community that can bring insights into the maneuver detection use-case, namely, the algorithms for detecting time-series motifs. Particularly, after a review of the available algorithms for motif detection, we made our own implementation of a specific algorithm, applied it to a publicly available naturalistic driving dataset and explored the relationship between the extracted motifs and maneuvers.

The rest of the paper is organized as follows - Section 2 includes a review on the methods for identifying driving behavior with telematic data and explores the field of time-series motif detection algorithms, Section 3 presents the specific algorithm used for this paper and explains the modifications added to better fit this use-case, Section

4 summarizes the main results and the fifth and final section concludes the paper and opens the path for future research.

3.2 State of the art

3.2.1 Driving behavior in telematic data

Identifying driving behavior or detecting maneuvers high-frequency trip recordings is not a straightforward task and many authors proposed some ways of tackling this problem. Nevertheless, most papers can be aggregated into two main strategies - fixed thresholds [12, 13, 14, 15] on inertial variables and rolling windows [16, 17, 18, 19, 20, 21, 22, 23, 24].

In the fixed thresholds strategy, authors set specific thresholds on acceleration or other inertial measurements in order to define the beginning and end of maneuvers. These maneuvers are then used as input to the driving performance algorithm. As an example, Paefgen et al. [13] defined four types of maneuvers by specifying thresholds on lateral acceleration (for the left turn and the right turn) and on longitudinal acceleration (for forward acceleration and for braking). In another paper, Johnson and Trivedi [12] applied a simple moving average on the rotational energy derived from smartphone's sensors and defined a specific interval to set the beginning and end of a maneuver, independently of its type. These types of approaches have the advantage of being easy to implement, lightweight and interpretable. However, since the choice of the thresholds is essential to the detection of maneuvers, these methods require fine-tuning for each specific dataset and are inflexible to changes in the data.

The second type of strategies involves the use of rolling windows, in which the trip is divided into fixed-sized time windows that can have some level of overlap. Then, each window is used as input to a supervised model that classifies it as a specific maneuver. In this strategy, the window size and the percentage of overlap are important parameters that need to be set beforehand. Saleh et al. [16] built windows from inertial sensors and classified them as aggressive, normal or drowsy using a stacked-LSTM model. Each window had 64 time steps and had a 50% overlap. On a different direction, Xie et al. [19] compared three feature extraction methods for identifying various maneuvers such as braking, turning and acceleration. They also used rolling windows for splitting the trips and they tested different window sizes with no overlap. The authors state that the window size is very relevant to the classification performance and that the optimal size can vary depending on the type of maneuver, which leads to the conclusion that an "adaptive window sizing method"would be better suited. Thus, again, the inflexibility of the window size is a disadvantage in this method as well.

Another issue with the rolling windows method is the overlap parameter. If no overlap is used, then the way a trip is split may not be optimal since a window can also split a single maneuver and information may be lost. On the other hand, using

the maximal overlap (i.e., moving the window a single time step) comes with its own problems. In their paper, Keogh and Lin [25] argued that clustering time-series subsequences is meaningless when rolling windows are used to build the subsequences. In particular, they state that *"clusters extracted from these time series are forced to obey a certain constraint that is pathologically unlikely to be satisfied by any dataset, and because of this, the clusters extracted by any clustering algorithm are essentially random"*.

It is important to note that Saleh et al. [16] used an 50% overlap with great success, which seems to validate the use of rolling windows. However, the authors used a supervised approach and Keogh and Lin [25] only argues against using rolling windows in unsupervised methods. Nevertheless, building high-frequency labeled datasets for driving behavior is quite laborious and requires a lot of resources, which is evident in the extremely low number of labeled datasets available to researchers. Thus, and since the possibility of using unsupervised or semi-supervised methods is quite desirable, for this specific problem, using rolling windows is not the optimal approach.

The paper from Keogh and Lin [25] not only proves the point of the meaninglessness of clustering rolling windows but also proposes a solution, *motif-based clustering*. According to the authors, time-series motifs are over-represented subsequences in a time-series. The concept of motifs was first introduced in the genetics research as sequences of amino-acids in the DNA with biological significance [67] and, since then, the field of motif discovery for time-series has been receiving a lot of attention from the data mining community. Keogh and Lin [25] proposes to use a motif detection algorithm to find the set of motifs with the highest representation in the original time-series and then to apply the clustering algorithm directly to the motifs.

Therefore, in this paper, we propose to use a motif detection algorithm in inertial measurements to extract maneuvers from a trip. In other words, our main hypothesis is that over-represented segments of inertial time-series are highly connected to maneuvers and, by looking at the most relevant motifs for a trip, we will be able to get insights on the maneuvers performed during that trip.

In the next subsection, we will explore the motif discovery task, taking into consideration the specific requirements of the maneuver detection use-case. The method needs to be flexible enough to detect maneuvers with variable sizes, which implies that the motif detection algorithm needs to be able to extract variable-length motifs.

3.2.2 Motif discovery overview

Even though the concept of motifs relates to over-represented time-series segments, the exact definition of the most relevant motifs varies slightly among authors and areas of application. Mueen [68] states that there are two main definitions, namely a similarity-based and a support-based. In the first, the motifs can be ordered based on the similarity of the segments that belong to the motif, while the second orders motifs based on the number of repetitions of the motif throughout the time-series. Thus,

the similarity-based definition results in highly similar motifs and the support-based definition results in highly frequent motifs.

Additionally, there is some constrains which the time-series segments belonging to a motif must meet [66]. The behavior constrain introduces the idea that segments in a motif should present the same general behavior. Note that, depending on the application, distortions such as noise or time and amplitude shifts may be accepted [67]. The distance constrain goes a bit beyond by stating that all motif's segments need to have a distance smaller than a predefined radius R to the center of the motif (i.e., the segment that represents that motif). This radius is a very important parameter in any motif detection algorithm and it needs to set beforehand. Finally, the non-overlapping constrain sets that motif segments cannot overlap in time, which avoids what Lin et al. [65] refers to as trivial matchings.

There are also some motif detection use-cases in which one is not interested in finding the single most important motif, but instead the k most important motifs or, as referred in the literature, the *k-motifs* [65]. Put simply, based on the desired motif definition, one can order the entire set of motifs found with a certain algorithm and then extract the first k ordered motifs. However, in order to avoid extracting motifs with overlapping members, these motifs have to meet a distance constrain, namely, all the pairs of motif's representative (or the centers) need to have a distance higher than $2R$, where R is the radius discussed in the previous paragraph.

Independently of the exact definition, motif discovery is a computational intensive task as it involves comparing all possible pairs of time-series subsequences. The most prevailing strategy to speed this search is to use a low dimensional representation of the original time-series in which the distance of subsequences in the original representation is approximately maintained [68]. This way, one is able to extract motif candidates in the low-dimensional representation (which is more efficient than use the original representation) and simply filter the candidates based on the real distance in the original representation. The most used representation for motif detection is the SAX, which transforms sliding windows of the original time-series into discrete sequences of characters [44]. Since each sliding window is encoded with a predefined number of characters, the continuous time-series is broken into a list of fixed-sized sequences of characters (or words). Another common approach is to used spectral representations such as the DFT, the classical representation of a time-series in the frequency domain [45].

Another important component of motif discovery is the distance used to compare time-series segments. Most papers use either the Euclidean distance [53] or the DTW [55]. While the first is very efficient and thus allows for a fast comparison of segments, the second has the advantage of allowing to compare segments with different lengths and with time distortions by computing the optimal alignment in time of the two time-series that are being compared.

Motif discovery in time-series started with the fixed-length motif problem, where

the goal was to find motifs with a predefined length. However, as stated in the introduction, we are interested in finding motifs with no predefined length and even allowing segments from the same motif to have different lengths. Therefore, because the fixed-length motif discovery is too restrictive for this use-case, we will need to use variable-length motifs.

According to Torkamani and Lohweg [67], there is a considerable set of papers that address the variable-length problem by applying a fixed-length algorithm to a range of window sizes and then choosing the most representative motifs based on their motif definition and motif ranking schemes [74, 75]. However, this strategy does not allow to have segments with different lengths in the same motif. On the other hand, Tanaka et al. [66] introduced a strategy that allows a more flexible comparison of motifs with different lengths. In particular, they proposed to use the SAX representation for building a discrete list of words and to aggregate repeating words in the same word.

3.3 Implementation of the motif detection algorithm

The motif discovery algorithm chosen for this paper is the EMD algorithm by Tanaka et al. [66]. The main advantage of this algorithm, and the reason why we chose it, is the possibility of having subsequences with different lengths in the same motif. The EMD algorithm has three main components:

- Discretization of the 1-dimensional time-series via an adaptation of SAX representation [44]. This adaptation is what allows the method to find segments with different lengths in the same motif. Put simply, the algorithm starts by applying the SAX representation to sliding windows of the same length, which produces a sequence of SAX words, each representing a specific window in the original time-series. Then, the algorithm looks for consecutive sets of equal SAX words. If consecutive repeating words are found, the algorithm aggregates them into the same modified SAX word and joins the windows that were represented by those repeated words. For instance, given the SAX sequence $C_{SAX} = [abc, abc, cde, fde, fde, fde]$ representing 6 sliding windows of size n, the adaptation returns the modified sequence $\tilde{C}_{SAX} = [abc, cde, fde]$ that represents 3 windows of size $n+1$, n and $n+2$, respectively.

- Extraction of all the variable-length motif candidates with an iterative pattern matching routine. This process iterates over all the sets of consecutive modified SAX words (or patterns) and looks for repeating patterns that meet the distance constrain discussed in previous section.

- Computation of the description length of each motif candidate, which is based on the Minimum Description Length (MDL) principle [76]. The description

length (or MDL cost) can be then used to select the most relevant motifs from the extracted candidates.

Since, to the best of our knowledge, there was not a publicly available implementation of the EMD algorithm, we did our own implementation of the algorithm [1]. In addition, implementing the algorithm gave us the the flexibility to add two simple features that were not discussed in the original paper and a major change that we will argue benefits the use-case of finding maneuvers from inertial measurements.

The first addition is related to the avoidance of the trivial matchings discussed in the previous section. In their paper, Tanaka et al. [66] did not give any details on how to prune overlapping subsequences that belong to the same motif. Therefore, we implemented it with a simple heuristic. Given a motif, its center subsequence and its members subsequences, when two members overlap, we exclude the member with the largest distance to the motif's center and keep the member with the lowest distance.

The second addition involves the extraction of k-motifs. Since we are interested in finding different maneuvers, we had to implement a routine that pruned the entire set of motifs found by the EMD algorithm in order to avoid overlaps between different motifs. Tanaka et al. [66] had already defined a method for ordering motifs using the MDL cost. However, they did not discuss the issue of overlapping motifs and thus we used the heuristic proposed by Lin et al. [65]. Namely, we defined the k-motif as the motif with the lowest MDL cost and whose center has a distance higher than $2R$ to the the center of each the j-motifs, for $1 \leq j \leq k-1$. We named this step *motif pruning*.

The major change we introduced to the EMD algorithm concerns the SAX representation. In the original method, each sliding window is transformed into a sequence of n numbers, which is defined by dividing the window into n equal-size segments and by computing the mean of each segment. Then, each of the n numbers is mapped to a unique character based a set of computed break-points. These break-points are defined separately for each window so that the characters' frequency in that window exhibit a Gaussian distribution.

With this adaptive break-points method, windows that have the same behavior but are shifted in amplitude are mapped to the same SAX word and consequently will be compared as candidates for the same motif. Although useful in some cases, because amplitude is extremely important for identifying different maneuvers, this adaptive feature is not desirable in our use-case. As an example, if we used adaptive thresholds, acceleration increases from 0G to 0.2G would have the same SAX word as acceleration increases from 0G and 0.8G and thus, we wouldn't be able to have these two subsequences as distinct motifs. In other words, we would not observe smooth maneuvers and aggressive maneuvers as different motifs.

Finally, after some investigation of the results, we observed that the motif punning reduced significantly the number of motifs detected. In order to better investigate all

[1] https://github.com/misilva73/manueverMotifs

the original motifs found by the EMD algorithm, we applied a clustering method to the centers of each motif and grouped them based on their DTW distance. For this purpose, we used the DBSCAN algorithm [77]. This is a density-based algorithm that defines core points of high-density (points with many close neighbors) and expands clusters from these points. The algorithm has two main parameters, namely, the maximum distance to consider two points neighbors and the minimum number of neighbors a points must have to be considered a core points. Based on this two parameters, the algorithms finds both outliers and the clusters in the data. This is a main advantage as we do not need to give a predefined number of clusters and can discover outlier motifs, which can be also interesting to analyze.

3.4 Results and discussion

In order to investigate whether the EMD algorithm is able to extract meaningful motifs for detecting maneuvers, we applied the algorithm to the UAH-DriveSet, a publicly available naturalistic driving dataset with trip recordings from six different drivers and two specific routes in Madrid, Spain. Romera et al. [26] asked each driver to repeat two predefined routes simulating three different behaviors, namely, normal, aggressive and drowsy. During the trips, they collected both raw and processed signals using an app designed by them and called DriveSafe [27, 28].

We run two experiments[2]. In the first, we aimed to identify brakes and accelerations, while in the second we focused on curves and other lateral maneuvers. For each experiments, we run the algorithm in four different trips - one representing a normal trip in a secondary road and three related to trips in a motorway and exhibiting the three behaviors present in the dataset (normal, aggressive and drowsy).

In the experiments, we used the accelerometer's measurements in the y and z axis with a frequency of 10Hz, which were already aligned with the three car axis and denoised with a Kalman filter. Since the y-axis represents the lateral acceleration of the car and the z-axis represents the longitudinal acceleration of the car, we used the first to detect curves and the second to detect accelerations and brakes. This dataset also included a record of the start of acceleration events captured by the DriveSafe app, such as turns, brakes and accelerations. These events were marked with a fixed-thresholds strategy and thus, we used these labels to help in the motif visualization.

The EMD algorithm has four main parameters, namely, the window size used to build each individual SAX word, the Piecewise Aggregate Approximation (PAA) size, which corresponds to the number of characters in each SAX word, the SAX alphabet size, which is the maximum number of characters used in the SAX representation, and the radius R, discussed in the definition of a motif. We tested a range of values for each parameter and, based on an exploration of the resulting motifs, the final parameters

[2] https://github.com/misilva73/manueverMotifs

used in all the experiments were `window_size = 20`, `paa_size = 2`, `alphabet_size = 5` and `R = 0.1`.

In order to validate the results, we used the video recordings of the trips and the measurements of car's velocity. The videos were mostly useful in the lateral experiment, while the velocity was used to validate the longitudinal maneuvers. Note that we did this in-depth analysis on the smaller set of pruned motifs as the full set of extracted motifs was too big for such manual analysis. Therefore, we applied the clustering algorithm DBSCAN [77] as an exploratory tool.

3.4.1 Identifying brakes and accelerations in the longitudinal acceleration

Table 3.1 summarizes the main results obtained with the EMD method and the DBSCAN clustering. For each trip, it displays 1) the number of motifs extracted by the EMD algorithm, 2) the number of motifs remaining after the motif pruning, 3) the types of maneuvers detected after punning, 4) the number of clusters obtained with the DBSCAN algorithm, excluding the outliers cluster, and 5) the number of motifs in the outlier cluster.

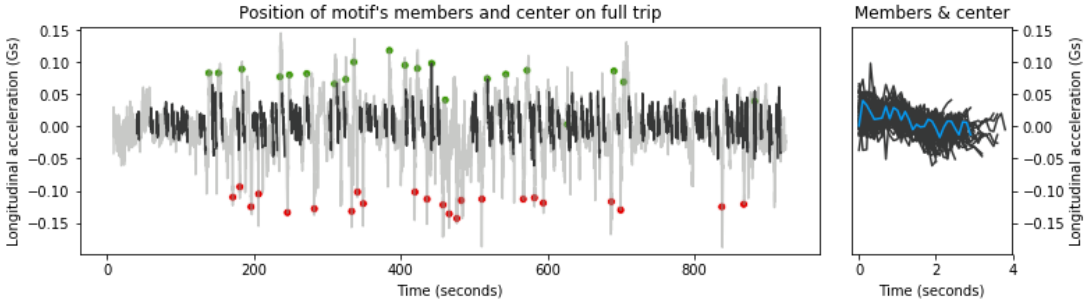
Table 3.1: Summary results of the analysis in the longitudinal acceleration

Trip		Motifs	Motifs after pruning	maneuvers[*]	DBSCAN clusters	DBSCAN outliers
Motorway	Normal	1849	3	B, B-A	5	383
	Aggressive	1272	8	B, B-A	8	771
	Drowsy	2039	5	B, B-A	5	477
Secondary	Normal	1591	3	A, B-A	2	162

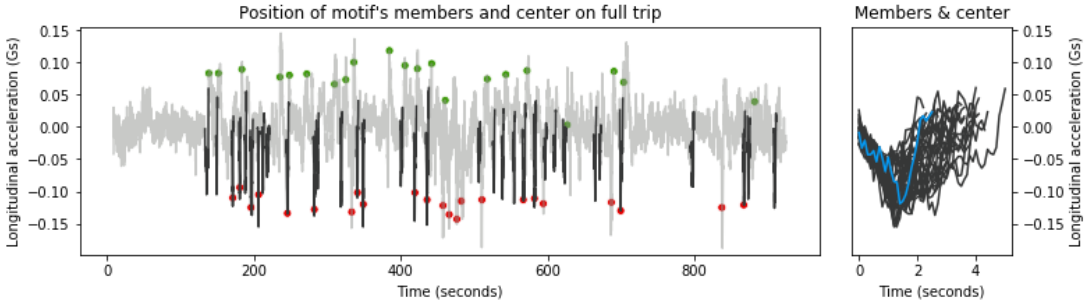
[*] A = Acceleration ; B = Brake ; B-A = Brake followed by acceleration

From these results, we were able to observe that the motif pruning step reduces the number of motifs by three orders of magnitude, which is a very significant reduction. This indicates that most of motifs detected by the EMD algorithm are in fact very close to each other and therefore represent either the same maneuver or very similar maneuvers. Additionally, after the motif pruning step, the remaining motifs in all trips contained relevant maneuvers, which indicates that motif detection algorithms can help in the task of discovering maneuvers in a signal of longitudinal acceleration. Figure 3.1 displays three motifs extracted with the EMD algorithm and they all have an easily identifiable maneuver.

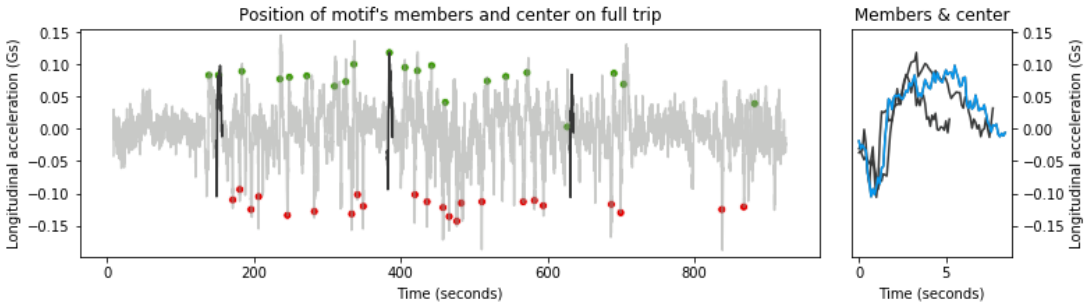
In all the trips explored, the most relevant motif found by the EMD algorithm contained the absence of a maneuver. In other words, this motif corresponded to the action of driving at a constant speed without changes in longitudinal acceleration, which is consistent with routes where these trips where recorded. In motorways and secondary roads, driving is done in a constant speed at most times and thus we expect to have this behavior as the most significant motif of these trips.



(a) No maneuver



(b) Brake maneuver



(c) Brake-acceleration maneuver

Figure 3.1: Three example motifs found in the motorway aggressive trip. They correspond to the first, second and fifth most relevant motifs, after pruning. Red and green points represent the original labels (computed using a threshold method) present in the UAH-DriveSet, where red marks indicate the beginning of a brake and green the beginning of an acceleration.

It is also interesting to note that in all the four trips, the algorithm was able to identify a motif with the maneuver of a brake followed by an acceleration, which could be indicative of a tailgating behavior.

Another common factor to all the explored trips was the variability of the lengths of segments belonging to the same motif, which was one of the main reasons for choosing the EMD algorithm in the first place. The ability of extracting motifs whose members have slightly different lengths is a major advantage of this algorithm as it allows us to find maneuvers that have the same type (e.g. a brake) but have lightly different lengths.

We also observed that this method was capable of finding subsequences that correspond to the same maneuver but, because they are slightly bellow the thresholds set in the DriveSafe app, they are not marked as a maneuver in the original dataset. As an example, Figure 3.2 shows the plot of two subsequences that belong to the same motif. The first was marked as a brake but the second, because it was bellow the threshold set by the app, was not marked as a brake. This motif was extracted from the drowsy trip in the motorway route.

The results obtained with the DBSCAN clustering algorithm show that the majority of the motifs are very close and represent the same behavior, the absent of a maneuver. However, all trips have a cluster of outliers, which are motifs that are different from the majority motifs and do not have a big enough neighborhood to form a cluster. These are the most interesting motifs from the perspective of maneuver detection. Therefore, investigating outlier motifs can be a way of enriching the motifs found with the pruned version of the EMD algorithm.

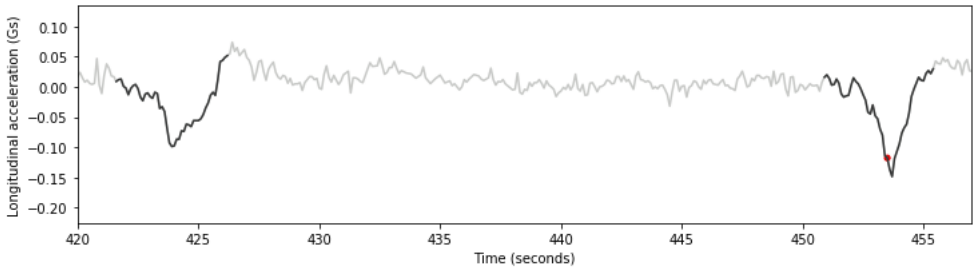


Figure 3.2: Zoom-in on a motif extracted from the drowsy trip in the motorway route. The second subsequence was marked as a brake in the UAH-DriveSet and the EMD algorithm extracted it as a member of a motif. The first, although not being marked, could be also recognized as a brake by belonging to the same motif as the first subsequence.

3.4.2 Identifying turns in the lateral acceleration

Similarly to the previous section, Table 3.2 summarizes the main results obtained with for the lateral acceleration analysis and Figure 3.3 shows three motifs extracted with the EMD algorithm with pruning. In general, results were consistent with the ones

obtained in the longitudinal acceleration experiments and thus conclusions are not much different.

Table 3.2: Summary results of the analysis in the lateral acceleration

Trip		Motifs	Motifs after pruning	maneuvers[*]	DBSCAN clusters	DBSCAN outliers
Motorway	Normal	1532	3	LC, OT	6	1328
	Aggressive	634	7	D, LC	1	614
	Drowsy	1464	8	LC, C	5	1292
Secondary	Normal	1113	1	None	2	929

[*] LC = Lane change ; OT = Overtaking ; D = Drift, C = Curve

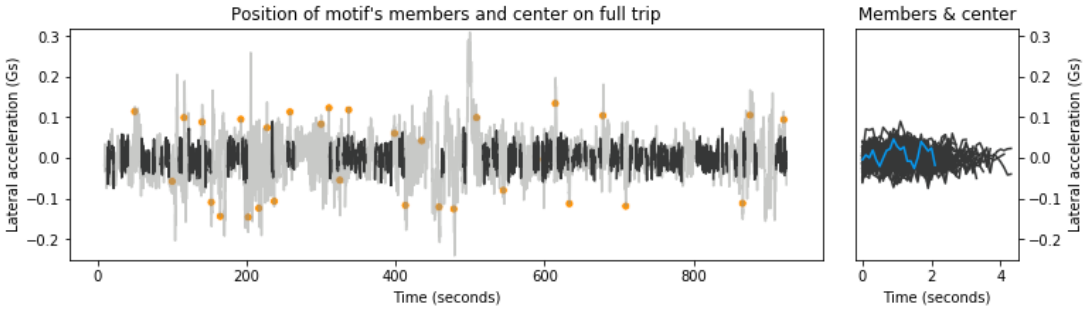
Motif pruning continues to reduce significantly the number of motifs and, after pruning, we could identify some relevant maneuvers in the remaining motifs. In all trips, the most relevant motif was the one representing the absence of a maneuver which was expected given that both routes do not have many curves. In three trips, some of the motifs extracted after the pruning related to lane changes, which was easily validated in the videos recordings. Additionally, in one of the trips, we were able of observe a motif that included a mush more complex maneuver, namely a overtaking maneuver.

The video recordings also showed that some motifs included clear patterns in the lateral acceleration which, in most cases, were related to a lane change but, in a few cases, included no visible maneuver. We hypothesize that these patterns in the acceleration time-series can be caused by road inclinations or blasts of wind. This means that by looking at a single signal, motifs will be more prone to errors and thus, as future work, we should combine multiple sensors and find multi-dimensional motifs.

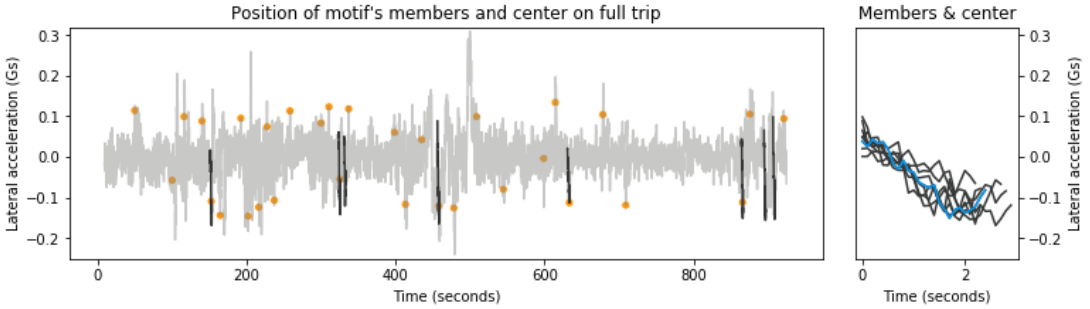
Compared to the longitudinal analysis, the DBSCAN detected more outliers, which suggests that this signal leads a higher number of motifs which are distant from the "no maneuver"behavior and thus one would need to spend more time analyzing these motifs.

3.5 Conclusion and future work

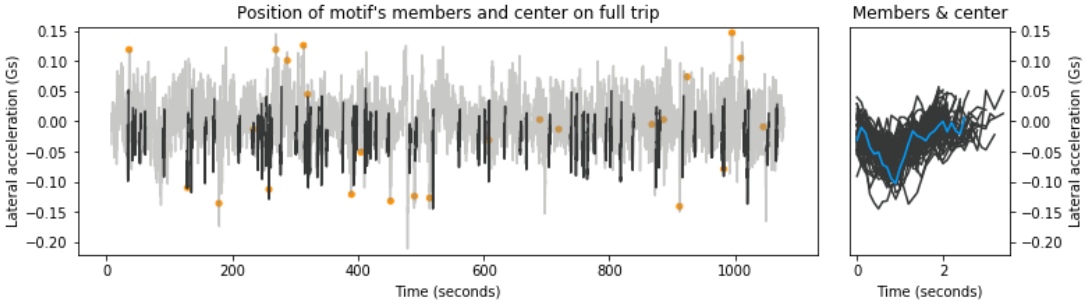
As many papers have shown, driving behavior has a great impact on road safety. A popular way of analyzing driving behavior is to move the focus to the maneuvers as they give useful information about the driver performing those maneuvers. In this paper, we investigated a new way of identifying maneuvers from vehicle telematics data, motif detection in time-series. Put simply, time-series motifs are over-represented subsequences in a time-series [65]. With the hypothesis that over-represented segments of inertial time-series are highly connected to maneuvers, we sought to analyze the relationship between the most relevant motifs of a trip and the maneuvers performed during that trip.



(a) No maneuver



(b) Lane change



(c) Drift

Figure 3.3: Two example motifs found in the motorway aggressive trip (1st and 4th motif) and one example found in the motorway drowsy trip (2nd motif). Orange points represent the original labels present in the UAH-DriveSet (and computed using a threshold method), where the marks indicate the beginning of a turn.

We implemented a slightly modified version of the EMD algorithm [66], a classical motif detection algorithm for time-series which is capable of finding subsequences with different lengths in the same motif, and we applied it to the UAH-DriveSet [26], a publicly available naturalistic driving dataset with trip recordings from six different drivers and two specific routes in Madrid, Spain. Particularly, we ran two different experiments. In the first, we aimed to identify acceleration and brakes from the longitudinal acceleration time-series and, in the second, we aimed to identify turns from the lateral acceleration time-series.

After a systematic exploration of the extracted motifs, we were able to conclude that the EMD algorithm was capable of extracting simple maneuvers such as accelerations, brakes and curves. We identified additional maneuvers that could be associated with more complex behaviors. In the longitudinal experiments, we identified a motif with the maneuver of a brake followed by an acceleration, which could be indicative of a tailgating behavior. In the lateral experiment, we found motifs with lane changes and overtaking maneuvers.

Additionally, by providing a way of ordering the motifs by its relevance in the trips, the EMD algorithm gives some extra information on the trip itself. As an example, in the drowsy trip, the second most relevant maneuver in the lateral acceleration was the drift while, in the aggressive trip, the lane changes occupied the most interesting motifs. This is very indicative of the type of driving that was being made in both trips: the aggressive trip was dominated by lane changes, thus showing a higher level of impatience, and the drowsy trip was dominated by drifts, which can be associated with inattention and sleepiness.

Although validating motif discovery as a worthwhile line of research for detecting maneuvers, there is still some work to be done. Firstly, the analysis performed in this paper was exploratory in nature as we did not have a more automated way of identifying which maneuver was being performed in each motif. Therefore, an important next step to make motif detection work in practice is to build a method for finding similar motifs and associating that motifs to a specific maneuver without the need to visual exploration.

Secondly, even though motif pruning showed some promise in ordering and selecting motifs, the DBSCAN analysis leads us to believe that by only looking at the pruned motifs and discarding all the others, we might be missing some interesting maneuvers. Therefore, it would be also interesting to further investigate a better grouping of the motifs as a way of finding more instances of with maneuvers.

Thirdly, we applied the EMD algorithm to single trips independently. This means that if a driver performs a specific maneuver only once during the trip, the EMD will not recognize it as a motif since it only appears once. However, if we were to expand the search for motifs to a varied set of trips instead of focusing on a single trip, we expect to see these cases become very rare. Thus, this work should be further extended to multi-trip motif discovery.

Finally, we used the lateral and longitudinal acceleration time-series separately to find simple maneuvers such as turns and brakes. However, the ideal setup would be to look for motifs in the two acceleration axis at the same time and also include other sensors such as velocity. This way, we would be able to extract more complex maneuvers and we would reduce errors created but changes in road inclination or blasts of wind.

This idea is equivalent to the task of detecting motifs in multidimensional time-series. The two main ways of tackling this problem are to reduce the number of variables to one and apply a motif discovery algorithm to the resulting 1-dimensional time-series [66] or to apply a motif discovery algorithm to each individual dimension and search for co-occurrences of the 1-dimensional motifs to extract the final multidimensional motifs [70, 71, 72]. While the first has the advantage of avoiding running a motif detection algorithm in all the dimensions, the second is more accurate as there is no information loss due to dimensionality reduction. Therefore, it would be very important to explore these two options and to discover if they are suited for the use-case of maneuver detection.

Chapter 4

Article no. 2 - Exploring time-series motifs through DTW-SOM

This chapter contains the second article that composes this thesis. The article was accepted for presentation at the International Joint Conference on Neural Networks (IJCNN) 2020 and published in the conference proceedings. The published version can be accessed at https://doi.org/10.1109/IJCNN48605.2020.9207614.

In the previous chapter, we notice that the motif pruning step from the EMD algorithm is too restrictive and thus excludes relevant maneuvers from the trip. On the other hand, the number of motifs extracted by the algorithm is too large to allow a human to explore it manually. In other words, we need a summarization and visualization tool to get the full benefit to motif detection algorithm for the use-case of maneuver detection.

With this in mind, in our second article, we proposed the use of the SOM clustering algorithm to explore the motifs extracted by any desired motif discovery method. This new algorithm is called DTW-SOM and it receives the centers of a set of motifs, groups them into a predefined set of clusters and provides a space-efficient visualization that aids the investigation of the main extracted patterns.

4.1 Introduction

In the last decade, motif discovery has become a fundamental step in many data mining tasks for time-series data, such as clustering, classification or anomaly detection. In general, a time-series motif corresponds to a over-represented segment of a time-series and thus motif discovery involves extracting all (or a specific subset) of these over-represented segments [68]. Figure 4.1 illustrates an example of two motifs built from dummy data.

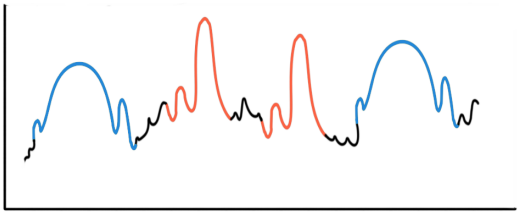


Figure 4.1: Toy example of two different motifs, each with two highly conserved subsequences.

Due to its relevance, many methods and strategies have been proposed to tackle motif discovery. However, the step of exploring and visualizing motifs, which can be useful to understand results of downward tasks, has not received as much attention. To the best of our knowledge, papers that address this question focus only on visualizing the actual time-series subsequences that belong to each individual motif [78, 79, 80]. We argue that, even though exploring individual motifs can help to understand the individual patterns, these methods cannot provide information about the overall relationships between the extracted motifs. In other words, they are not ideal to answer questions such as: Are motifs similar to each other? Can we define clusters of motifs? Additionally, exploring individual clusters is not tractable in the cases where a high number of motifs is extracted.

In this paper, we propose the use of a widely-studied method for feature reduction and visualization, the SOM [81], to explore the centers of motifs extracted by any desired motif discovery algorithm. Taking into account that these centers are time-series subsequences, with possibly variable lengths and multiple dimensions, we adapted the original SOM algorithm to apply the DTW distance [55] as its similarity metric and added two specific initialization routines for the SOM network.

The rest of the paper is organized as follows: Section 2 introduces academic work related to (1) motif discovery and (2) Self-Organizing Maps, Section 3 describes our own implementation of the DTW-SOM, Section 4 presents the experimental setup and reports the results obtained on three different datasets and, finally, Section 5 concludes this paper and discusses future work.

4.2 Related work

In this section, we'll cover two areas which, although seemingly unrelated, serve as basis for this paper - motif discovery and self-organizing maps.

4.2.1 Motif Discovery

Despite, in general, the concept of motifs being associated with significant time-series segments, there are two main definitions of motifs that vary on the way they set the concept of "significance"[68]. Similarity-based motifs focus on the similarity of the time-series segments and thus this definition results in highly similar motifs. On the other hand, support-based motifs focus on the repetition of the segments throughout the time-series and thus this definition leads to highly frequent motifs.

In addition to the concept of significance, there are additional constraints a group of time-series segments must meet to be considered a motif [66]. The first is a behavior constraint, which determines that segments in a motif should have the same general behavior, even if some level of noise or time and amplitude shifts are allowed [67]. The second, the non-overlapping constraint, aims to avoid trivial matchings [65] by setting that motif segments cannot overlap in time. The third is a distance constraint, which restricts all motif's segments to have a distance smaller than a radius R to the center of the motif (i.e., the segment that represents that motif). Note that in most motif discovery methods, the radius is a parameter that needs to be set by the user. Finally, when the task is to extract a set of motifs instead of the most significant motif, there is an additional constraint with the goal of avoiding the extraction of motifs with overlapping members. This last constraint states that all the motifs' centers must have a distance higher than 2 radius, $2R$.

Independently of the exact definition, motif discovery is a computationally intensive task as it involves computing distances between all possible pairs of time-series subsequences. Therefore, much of the work related to motif discovery algorithms has aimed at making the search more efficient. One of the most common techniques to reduce the search space is to convert the original time-series into a lower-dimensional representation where the distance between subsequences in the original representation is approximately maintained [68]. With this strategy, motif candidates can be extracted in the low-dimensional representation (which is more efficient) and the final set of motifs can be computed with the real distance on the smaller set of candidates. In motif discovery, the SAX representation [44] is the most used. This representation first extracts sliding windows from the original time-series and then converts each sliding window into a fixed-sized sequence of characters. Thus, the time-series is split into a list of "words".

An important part of any motif discovery is the choice of distance. Most authors use the Euclidean distance [53] or the DTW [55] for comparing subsequences. The

advantage of the Euclidean is its efficiency, which allows a faster comparison of motif candidates. However, this efficiency comes with a loss of flexibility as it is not robust to time-shifts, distortions, differences in phase and variable-length sequences. On the other hand, the DTW distance finds the optimal time alignment between the subsequences that are being compared. Thus, it is much less efficient but it can adapt to the shifts discussed before.

In terms of the algorithms for motif discovery, there are two main types, namely fixed-length and variable-length. In the fixed-length algorithms, users need to provide the length of window and that parameter is used to extract all the possible subsequences. Thus, all motifs contain subsequences of the same size. The MK exact algorithm [82], the motif extraction from the Matrix Profile [69] and the EMMA-SAX algorithm [65] are all examples of methods that extract fixed-length motifs.

The variable-length algorithms are a bit more flexible and don't required the user to set the window size beforehand. However, this is a much harder problem as the search space is bigger and more complex. Most variable-length algorithms solve this problem by applying a fixed-length algorithm to a range of window sizes and then choosing the most representative motifs based on their motif definition and motif ranking schemes [67]. The work of [74] and [75] are two examples of this approach. On the other hand, Lin's grammar-based approach [83] and Tanaka's EMD algorithm [66] already take into consideration variable-length motif during the search process by adapting the way subsequences are represented in the low-dimensional representation.

4.2.2 Self-organizing map

SOM were proposed by Tuevo Kohonen at the beginning of the 1980s [81], and constitute the product of his work on associative memory and vector quantization. The SOM's basic idea is to map high-dimensional data onto a low-dimensional discrete feature map, maintaining the relations between data patterns [84]. Its main objective is to "extract and illustrate"the essential structures from a dataset through a map resulting from an unsupervised learning process [85, 86] and thus it can be used at the same time for visualization and exploration of data and for clustering.

SOM is also considered a good method for extracting data patterns and associations when the extraction of information becomes a challenging task due to the number of parameters or the use of a multidimensional dataset.

Usually, SOM maps the original high-dimensional data to a discrete feature map with one, two, or three-dimensions, although 2-dimensions are the most common. The grid formed by the units or neurons is usually referred to as output space, as opposed to input space, which is the original space [87]. When the output space is 2-dimensional, it is usually formed by a rectangular or hexagonal grid of units [81].

Each unit of the SOM, is represented by a vector $m_i = [m_{i1}..., m_{in}]$ of dimension n, where n equals the dimension of the input space. In the training phase, a given

training pattern x is presented to the network, and the closest unit is selected. This unit is called the best-matching unit (BMU). The unit's vector values and those of its neighbors are then modified in order to get closer to the data pattern x:

$$m_i = m_i + \alpha(t)h_{ci}(t)\|x - m_i\| \tag{4.1}$$

where $\alpha(t)$ is the learning rate at time t, and $h_{ci}(t)$ is the neighborhood function centered in unit c, and i identifies each unit. To allow SOM to converge to a stable solution, both the learning rate $\alpha(t)$ and the neighborhood radius $h_{ci}(t)$ should decrease to zero during training. Usually these parameters decrease in a linear fashion but other functions can be used. Additionally, the update of both parameters can be done after each individual data pattern is presented to the network (iteration) or after all the data patterns have been presented (epoch). The former case is known as sequential training and the latter is usually known as batch training. The sequential algorithm pseudo-code is presented bellow, in Algorithm 1.

Algorithm 1: SOM Sequential Training

Input: $X = \{x_1, x_2, \ldots x_n\}$: training patterns
$W = \left(w_{ij}\right) \in \mathbb{R}^{p \times q}$: SOM network's units
$\alpha \in]0, 1[$: initial learning rate
$r \in \mathbb{R}$: initial neighborhood radius

Let $h(w_{ij}, w_{mn}, r)$ be the neighborhood function
repeat
  for $k = 1$ *to* n) **do**
    forall $w_{ij} \in W$ **do**
      $d_{ij} = \|x_k - x_{ij}\|$
    Select the unit that minimizes d_{ij} as the winner w_{win}
    Update $w_{ij} \in W : \; w_{ij} = w_{ij} + \alpha h(w_{win}, w_{ij}, r)\|x_k - w_{ij}\|$
  Decrease the value of α and r
until $\alpha = 0$;

For each randomly selected training pattern presented to the network, the BMU (i.e. its closest unit) is found. The BMU is then updated according to the weights of the training pattern and the learning rate. Initially this learning rate is high allowing bigger adjustments of the units. The unit's mobility will decrease proportionality with the decrease of the learning rate. Based on the neighborhood rate, a group of surrounding units is also moved closer to the training pattern. There are several ways to visualize the SOM and improve the understanding of the data patterns [88]. Two of the most important visualization tools are the component planes [81] and the U-matrix [89].

In a component plane, each unit is colored according to the weight of each variable in the SOM. Through the analysis of the component planes, data distribution can be evaluated. For instance, it is quite simple to identify variables which are correlated

(their component planes will have the same shape), and it is also possible to have an improved understanding of the contributions of each variable to the SOM. By comparing two or more component planes, one can visually identify correlations between variables, both globally and at a local scale.

The U-matrix is one of the most used methods to visualize SOM [89]. U-matrices are computed by finding the distances in the input space of neighboring units in the output space. There are two ways to visualize a U-matrix. The most common is to use a color code to depict distances, corresponding to the values of the U-matrix. Usually a grey-scale is used, with the highest value being represented with black and the lowest with white. Another possibility is to plot these distances in the form of a 3D landscape with mountains and valleys. A mountain region indicates large distances between units, while low distances between the units form valleys.

Some of the challenges in applying SOM to motif discovery in time-series are related to the fact that not all dimensions are equally relevant and the different size of motifs. Some work has been developed to overcame these problems [90, 91, 92]. One example is the Local Adaptive Receptive Field Dimension Selective Self-organizing Map (LARFDSSOM), proposed in [91], where the application of different weights for each input dimension and for each cluster is proposed, as well as the use of a Time-Varying Structure of the SOM. In [92] the authors propose VILMAP to allow the use of different sizes of samples and consequently the discovery of Motifs with different lengths.

SOM can, as shown in the previous references, be used to identify and extract time-series motifs. In this paper, we propose the use of this technique not to perform such tasks but to analyze the motifs extracted by others motif discovery algorithms.

4.3 DTW-SOM

As explained in previous sections, the goal of the DTW-SOM method is to serve as an auxiliary tool in motif discovery and, as such, it needs to be flexible enough to deal with a different range of motif discovery methods and definitions. Particularly, it should:

1. Be able to compare time-series segments.

2. Receive all types of motifs, meaning fixed-length and variable-length motifs.

3. Represent multi-dimensional motifs (i.e., motifs extracted from a multi-dimensional time-series).

4. Take into account an given order of significance in the motifs of a provided set of "most significant"motifs.

5. Provide a visualization from which a user can get insights into the general relationships between the motifs and their shapes.

To achieve these goals, we extended the vanilla batch-training SOM algorithm that was implemented in the PyClustering python package [93] to process variable-length multidimensional time-series subsequences. The idea is that when someone wishes to investigate a set of motifs previously extracted, he just needs to collect the center subsequences of each motif and feed that list of centers to the DTW-SOM. For readability, from now on, we'll refer to these center subsequences that are the input to the DTW-SOM as *patterns*.

Firstly, we added two network initialization routines, namely a random sample initialization and an anchor initialization. In the random sample initialization, each unit is randomly assigned to one input pattern in such a way that no two units are exactly the same. In other words, a random sample is taken from the input patterns to initialize the network. In the anchor initialization, a list of patterns must be provided by the users (the anchors) and each anchor will be set to a single unit. Note that the provided list cannot have more anchors than units and, in the case of an input with less anchors then units, a random sample will be taken from all the input patterns to initialize the remaining units. In the case of having less anchors than units, we tried to spread out the first anchors (as we are assuming that they are ordered by significance) by first filling the diagonals of the network and just after that filling the rest of the network.

Note that the anchor initialization was designed to address the fourth requirement presented above. If the motif discovery method provides an ordering of the motifs or a subset of most significant motifs, one can use these motifs as the anchors and thus the DTW-SOM will more easily focus on these more important motifs.

The second adjustment we implemented to the vanilla SOM was the swap of the distance function from the Euclidean distance to the DTW distance, which allows us to process patterns with different lengths and with multiple dimensions. Finally, the last big change implemented was in the *Adaptation* phase of the training. Since we are comparing patterns and units with variable lengths, we had to adapt the way each unit's values are updated. When computing the distance between two segments, the DTW distance finds an optimal alignment in time between the segments and uses this optimal match to compute the distance between individual points along the segments. Thus during the adaptation phase of training we leverage this matching to guide the update of the BMU's vector values.

As an example, Figure 4.2 shows the DTW alignment between an input pattern and its BMU. In the DTW-SOM, the vector values of an unit are simply the sequence of time-series observations as presented in figure. During *Adaptation*, every vector value needs to move slightly closer to the pattern as defined by equation 4.1. In this case, the

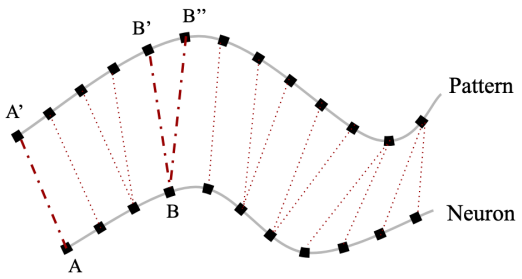


Figure 4.2: Example of the time-alignment between an input pattern and an unit computed by the DTW algorithm. The black points are the actual time-series observations while the red and dashed lines represent the alignment matching returned by the DTW algorithm.

first BMU's value, which is the point marked as A, will move closer to the its matching point in the pattern, which is the point A'.

Due to warping, we can have cases where one point in the BMU can be matched to more than one point in the pattern, as is the case with points B, B' and B''. In these cases, equation 4.1 needs to be changed to equation 4.2:

$$w_i = w_i + \alpha(t)h_{cw}(t)\left(\frac{1}{n}\sum_{j\leq n} x_j - w_i\right) \tag{4.2}$$

where i is the index of the unit's sequence, t is the training epoch, $\alpha(t)$ is the learning rate at epoch t, $h_{cw}(t)$ is the neighborhood function, n is the number of pattern points that where matched to the BMU's vector value w_i and the x_j are the pattern point's values.

4.4 Evaluation

To test DTW-SOM, we did three experiments. In the first, we generated synthetic motif data and explored visually the results. In the second, we used a widely used classification dataset from the UCR Time Series Classification Archive [94], the *GunPoint* dataset [95]. Particularly, we adapted the dataset to be suitable for the motif discovery task, we used the Matrix Profile [69] to efficiently extract all the motifs and we explored the resulting motifs with our method. Finally, in the third experiments, we used the same approach as in the second experiments, but used a dataset from the UCR Time Series Classification Archive with more classes and more extracted motifs, namely the *UWaveGesture* dataset [96].

4.4.1 Experiment with the synthetic motif dataset

To build this synthetic dataset, the idea was to create a dataset of motifs centers that formed 3 clear clusters. If we were able to detect these clusters in the final visualization,

then the DTW-SOM was working as expected. The synthetic dataset included 180 motif centers which were generated using the following heuristic:

1. We chose three general behaviors sequences for the clusters, namely, low-middle-high, high-middle-low and middle-middle-middle.

2. For each behavior (low, middle and high), we defined intervals from which we could sample points exhibiting that behavior. Particularly, the low interval was $[-3, -1.5]$, the middle interval was $[-0.5, 0.5]$ and the high interval was $[1.5, 3]$.

3. For each motif center, we set the length of which behavior by randomly selecting an integer between 5 and 10. In other words, for each motif center, we'll sample three integers that define the lengths of each of its behavior subsequences and the sum of those integers will be the total length of that motif center.

4. For each motif center, we create its time-series sequence by sampling values from the predefined behavior intervals. As an example, if we were creating a motif center for the low-middle-high cluster and if we had previously sampled the behavior lengths $(3, 7, 4)$, then we would sample three values from the interval $[-3, -1.5]$, seven values from the interval $[-0.5, 0.5]$ and four values from the interval $[1.5, 3]$.

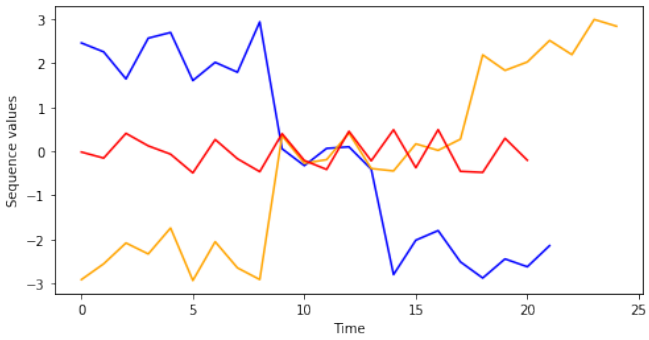


Figure 4.3: Plot with three examples of generated motif centers, one for each cluster. The orange belongs to the low-middle-high cluster, the blue to the high-middle-low cluster and the red to the middle-middle-middle cluster

Figure 4.3 shows some examples of the generated sequences. After generating the dataset, we build a DTW-SOM network with a $3X3$ layout and default parameters and we trained it during 30 epochs. We also tested the random sample initialization and the anchor initialization (using one motif from each cluster as the anchors). We noted that even though the random sample initialization was able to obtain the desired clusters, the results among different training runs were much more unstable. On the other hand, the anchor initialization converged to the desired clusters much more consistently and different runs did not change too much the results.

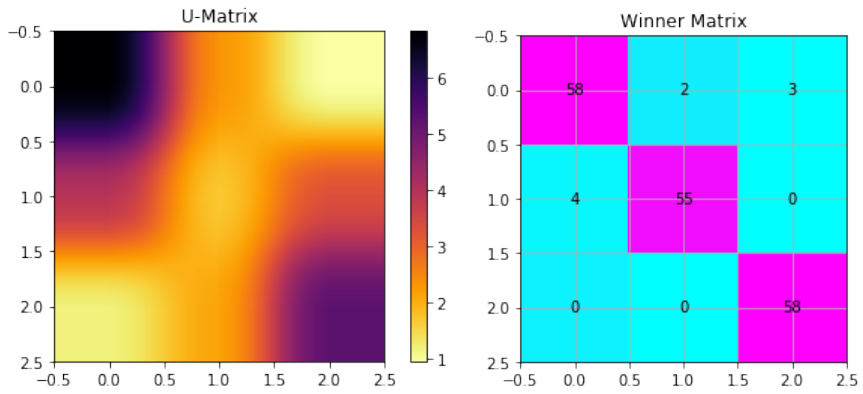


Figure 4.4: U-matrix and Winner Matrix obtained from the DTW-SOM trained on the synthetic motifs and using an anchor initialization.

Figure 4.4 shows the U-Matrix and the Winner Matrix obtained with the anchor initialization. Note that the winner matrix only encodes the number of input patterns that had each units as its BMU. We also plotted the sequence values of the nine units, which can be observed in Figure 4.5. From both plots, we can see that the diagonal of the network captures almost all the motifs as expected and that the rest of the units capture some other patterns between the middle-middle-middle cluster and the low-middle-high cluster.

4.4.2 Experiment with the *GunPoint* dataset

The *GunPoint* dataset [95] was built from the hand motion of two actors that are performing two different actions - the first is to draw a gun (which corresponds to the class "Gun") and the second is to point a finger (which corresponds to the class "Point"). The time-series correspond to the measurements in the x-axis of tracking the centroid of the actor's right hand. Because the setup was a classification task, we have a train and a test sets with 50 and 150 time-series sequences, respectively. Each time-series sequence includes the whole action of either the gun draw or the finger pointing and every sequence has a length of 150.

Due to the nature of this dataset, we had to adapt it to the task of motif discovery. For simplicity, we concatenated the 50 sequences from the train set into a single time-series and use this time-series as input to the motif discovery algorithm. Figure 4.6 includes a visualization with the original sequences and a subset of the final concatenated time-series.

The algorithm we used was the one based on the Matrix Profile [69]. Firstly, since we had the original lengths of the sequences, we could pose the problem as a fixed-length motifs discovery. Secondly, the Matrix Profile is known to be very efficient and, thirdly, its parameters are few and easy to tune.

Note that this method expects to receive as input the max number of motifs to find and thus this is actually a k-motif algorithm. However, if we set this parameter larger

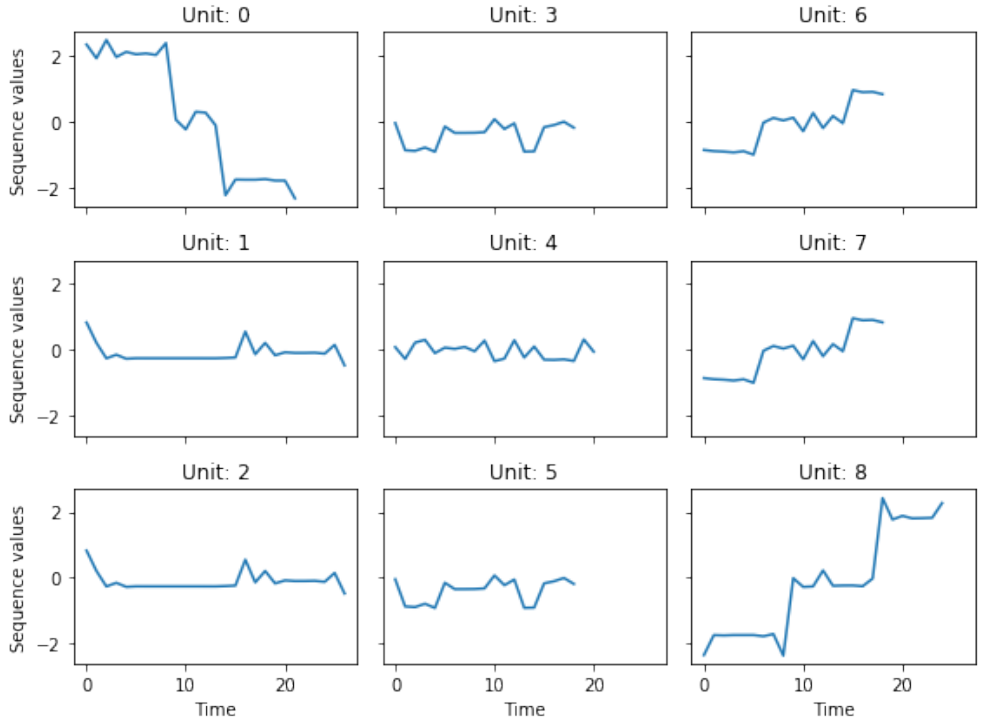


Figure 4.5: Sequence values of the units after training the DTW-SOM with synthetic motifs. The position of each unit's plot in the grid is consistent with its position in the network. Thus, the U-Matrix and the Winner matrix plots are consistent with this grid plot.

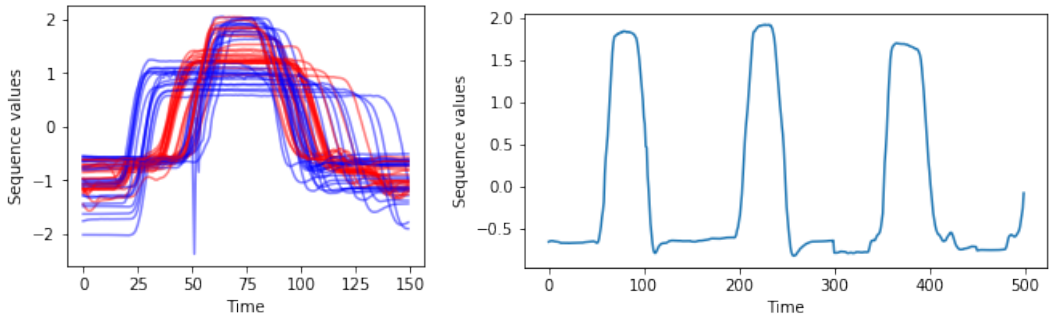


Figure 4.6: **Right plot:** original sequences from *GunPoint*'s train set. The colors indicate the different labels, "Gun"and "Point". **Left plot:** first 500 observations of the time-series built as a concatenation of the fixed-size sequences from *GunPoint*'s train set.

than expected (e.g. 1000 in our case), then the algorithm will return all the motifs it can find. This is exactly what we did and the algorithm managed to extract 25 motifs.

Finally, we built a DTW-SOM network with a $3X3$ layout, default parameters and a random sample initialization. We then rained that network with the list of the motifs' centers during 50 epochs. Figure 4.7 shows the U-Matrix and the Winner Matrix obtained from this DTW-SOM network and Figure 4.8 has the plots of the units' sequence values.

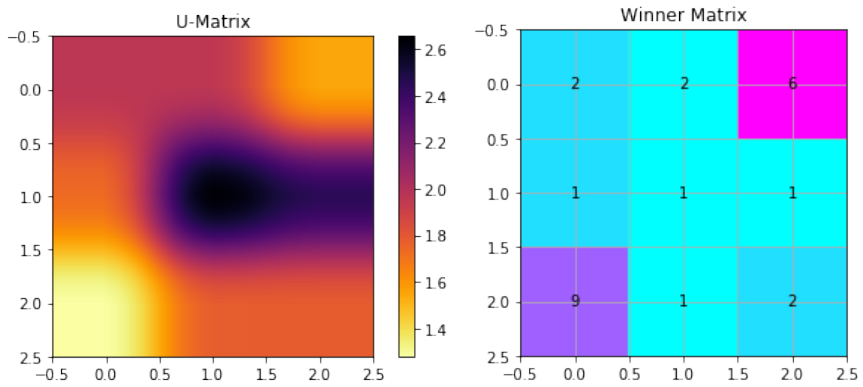


Figure 4.7: U-matrix and Winner Matrix obtained from the DTW-SOM trained on the motifs computed from the concatenated time-series of *GunPoint* sequences.

In this dataset, we can observe that DTW-SOM was capable of extracting some interesting information about the original 25 motifs computed with the Matrix Profile algorithm. Firstly, we can see two clear clusters around the units 2 and 6 (i.e. in the down-left and the up-right corners of the network).

Unit 2 corresponds to a pattern of raising the hand (either with a gun or not) and lowering the hand. Its neighbors, units 1 and 5, have the same pattern and since they were the BMU of a single input motif, they are essentially a cluster with unit 2.

Unit 6, on the other hand, has the pattern of lowering the hand and raising it again. It corresponds to the end of one of the original sequences and the start of the next one. Units 3 and 7 also have the lowering-raising patterns, however unit 3 has more time of the raising while 7 has more time of lowering.

Finally, units 0, 4 and 8 have their own specific pattern. Unit zero seems to have the original sequences of raising the hand a bit lower than the rest of sequences. In other words, they are the flat sequences in figure 4.6 that peak at the value 1. Unit 4 is the BMU of a single motif and encodes the "no action"pattern. Unit 8 has a quicker lowering pattern and thus encodes the end of the sequences with a quicker movement.

4.4.3 Experiment with the *UWaveGesture* dataset

The *UWaveGesture* dataset [96] corresponds to accelerometer recordings of right-hand gestures performed with the Wii remote. The dataset was built from eight participants doing eight specific gestures, which are presented on the left side of Figure 4.9. The

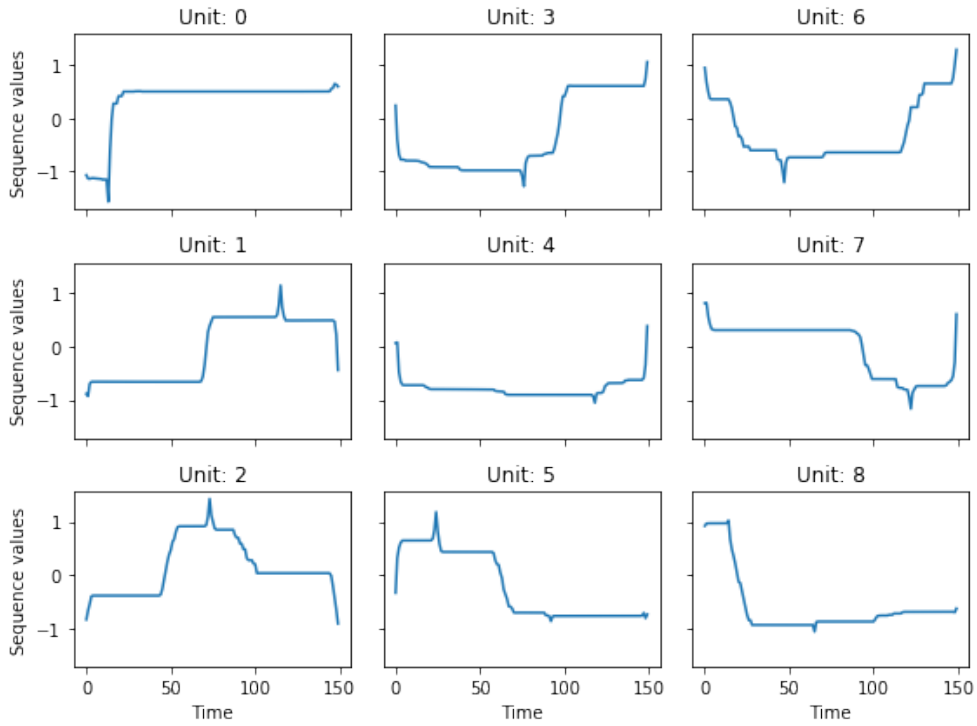


Figure 4.8: Sequence values of the units after training the DTW-SOM with motifs from the *GunPoint* dataset. The position of each unit's plot in the grid is consistent with its position in the network. Thus, the U-Matrix and the Winner matrix plots are consistent with this grid plot.

remote collects acceleration measurements from its three axis, as presented in the right side of Figure 4.9, and the UCR Time Series Classification Archive has one time-series dataset for each axis.

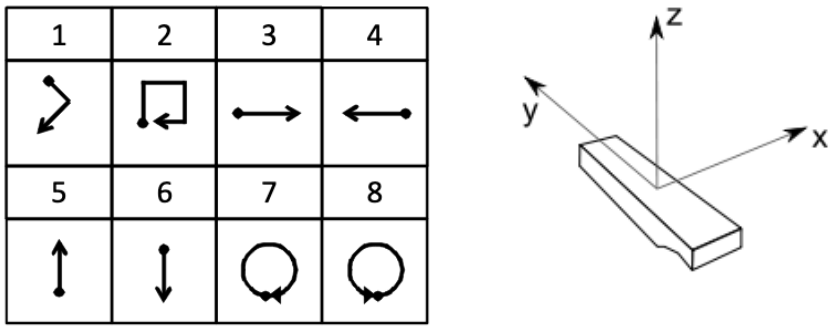


Figure 4.9: **Right plot:** Gesture vocabulary and related labels used in the *UWaveGesture* dataset, as presented in [96]. **Left plot:** Positioning of the accelerometer axis in the Wii remote.

For this experiment, we chose to use the dataset with the x-axis recordings, which corresponds to the lateral movements of the Wii remote. Because in this axis the gestures 5 and 6 have a zero acceleration (and thus are only noise), we excluded the

subsequences with these classes. In order to accelerate computation, we also sampled 400 sequences from the train set. Finally, similarly to the previous experiment, we concatenated the sampled sequences into a single time-series from which we could extract motifs. Figure 4.10 contains the original sequences, split by the gesture, and a subset of the time-series that resulted from the sequences' concatenation.

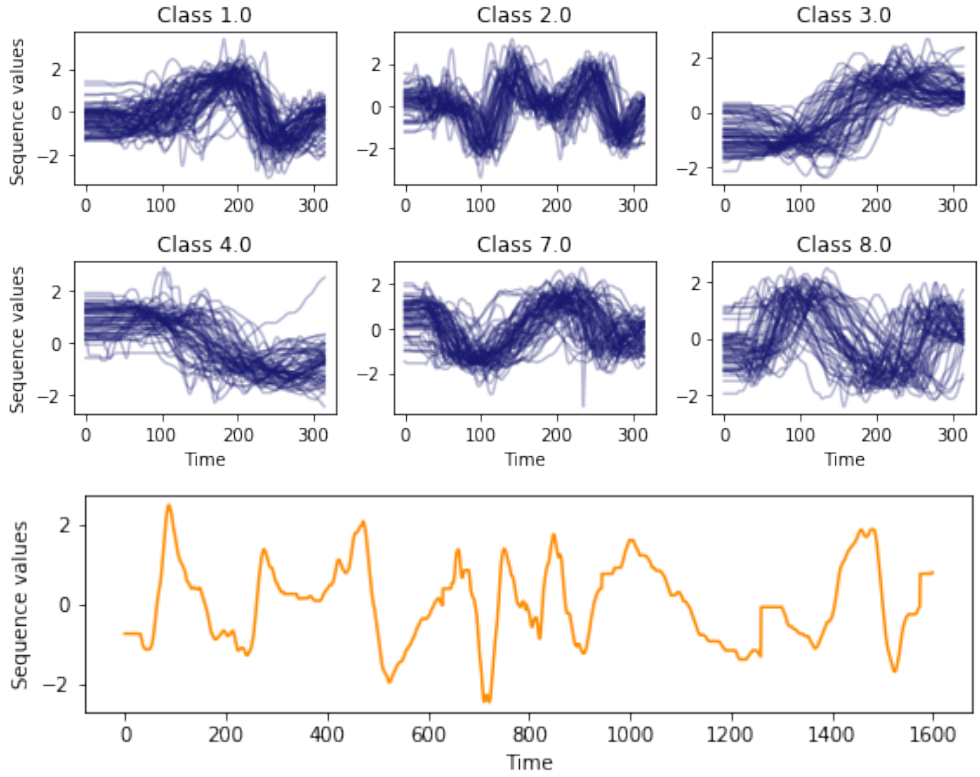


Figure 4.10: **Blue plots:** Original sequences sampled from *UWaveGesture*'s train set, split by the gesture class. **Orange plot:** First 1600 observations of the time-series built as a concatenation of the fixed-size sequences from *UWaveGesture*'s train set

In this experiment, we used again the Matrix Profile [69] with a larger than expected max number of motifs in order to extract all the fixed-length motifs. In this dataset, the algorithm extracted 125 motifs. We then trained a DTW-SOM network with a 4X4 layout, using the default parameters, a DTW maximum window of 100 (to limit the warping level) and a random sample initialization, which resulted in the U-Matrix, the Winner Matrix and the units shown in Figure 4.11 and 4.12.

From the U-matrix we can distinguish different regions of the DTW-SOM network. Unit 0, which forms its own cluster, has a simple shape similar to the third gesture. This is a simple gesture of moving the Wii remote to the right. Units 12 and 13 are far from their neighboring units and have a shape similar to the fourth gesture, or the gesture of moving the Wii remote to the left. These are the simplest gestures involving lateral movement of the Wii remote and it is expected that our motif detection algorithm would pick on these shapes.

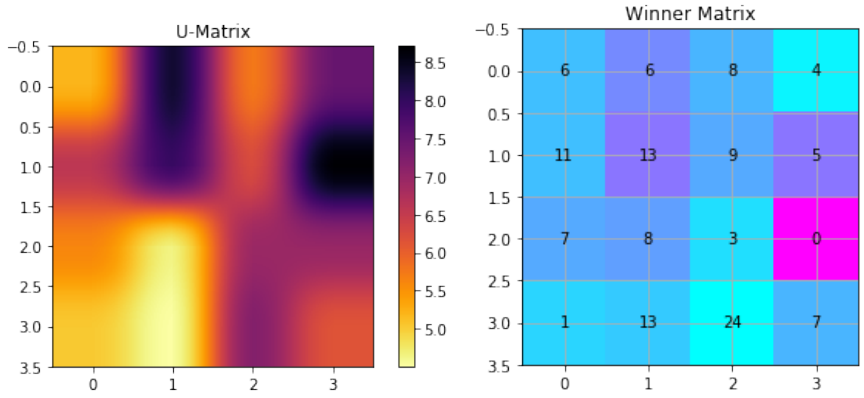


Figure 4.11: U-matrix and Winner Matrix obtained from the DTW-SOM trained on the motifs computed from the concatenated time-series of *UWaveGesture* sequences.

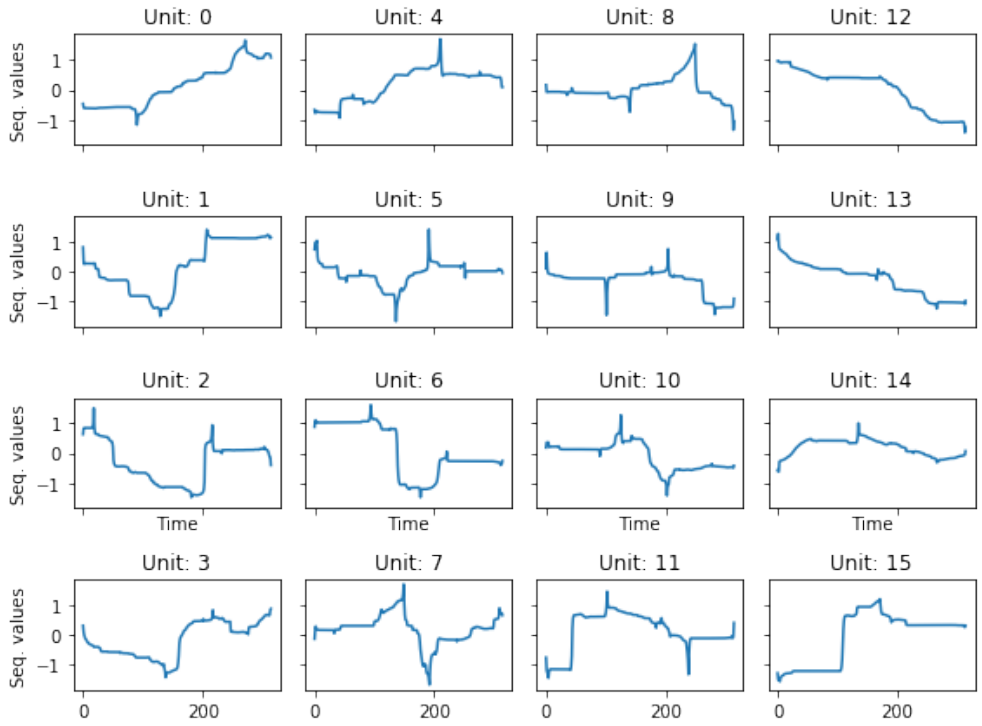


Figure 4.12: Sequence values of the units after training the DTW-SOM with motifs from the *UWaveGesture* dataset. The position of each unit's plot in the grid is consistent with its position in the network. Thus, the U-Matrix and the Winner matrix plots are consistent with this grid plot.

The visually biggest cluster in the U-matrix is centered at units 6 and 7. These units have a shape similar to the first and eighth gestures. These gestures are made of a right lateral movement followed by a left lateral movement. And so we can see that the motif detection algorithm is being capable of extracting more complex shapes.

Even though they don't form a visually striking cluster in the U-matrix, units 1, 2 and 5 have very similar shapes and are the only units with a shape consistent with the left-right lateral movement present in the seventh gesture.

Interestingly, unit 15 forms its own cluster in the lower-right corner of the DTW-SOM network, but it has a shape consistent with the third gesture. In other words, we have two clusters in the network, one in unit 0 and another in unit 15, with the same shape. This is due to the random initialization. Because these two similar motifs were randomly assigned to far away places in the network, they had no option but to form two independent clusters.

4.5 Conclusion

In this paper, we argue that visually exploring the time-series motifs computed by motif discovery algorithms can be useful to understand and debug results. To the best of our knowledge, no other papers investigate the problem of exploring relationships between motifs and answering questions such as: Are motifs similar to each other? Can we define clusters of motifs?

To conduct these investigations, we propose the use of an adapted Self-Organizing Map on the list of motif's centers. We called the adapted method DTW-SOM and the main changes are (1) the use the Dynamic Time Warping distance to compute distances between the units and the input patterns, (2) the introduction of two new network initialization routines and (3) the adjustment of the *Adaptation* phase of the training to work with variable-length time-series sequences.

We tested DTW-SOM in a synthetic motif dataset and two real time-series datasets called *GunPoint* and *UWaveGesture*, respectively. From an exploration of results, we can conclude that DTW-SOM is capable of extracting relevant information from a set of motifs and display it in a space-efficient way. During the experiment with the synthetic dataset, we observed that the random sample initialization was not as robust as the anchors initialization. Additionally, this random initialization can also lead to the creation of distinct clusters that have the same shapes, which is not optimal. Thus, as future work, we propose an investigation on more robust initialization schemes to cover the case when the user does not wish to provide anchors.

Chapter 5

Article no. 3 - TripMD: Driving patterns investigation via Motif Analysis

This chapter is made of the third and final article published for this thesis. The article was published in the journal *Expert Systems with Applications* and can be accessed at https://doi.org/10.1016/j.eswa.2021.115527.

the last article, brings together the work done in the previous two articles, which are located in Chapters

The main contribution here is the new system TripMD. This is the first that extracts the main driving patterns using motif detection and provides a summary of these patterns in a space-efficient visualization. With this, TripMD allows for an easy investigation of the main driving patterns obtained from high-frequency telematics data.

In addition, to make TripMD work better for maneuver detection, we also propose a new representation method for variable-length time-series called VSAX.

5.1 Introduction

In the last two decades, there has been a growing interest in analyzing driving data and understanding driving behavior, with researchers and practitioners finding new applications for this type of data. In the car insurance sector, measuring how a client drives is a cornerstone of the UBI schemes, which provide more custom pricing by taking into account driving behavior instead of external proxies such as sex and years of driving experience. In fleet management and fuel consumption optimization, studying the relationship between driving behavior and fuel consumption can improve driving performance and reduce costs. Regulators and policymakers can also leverage driving data to understand which factors are associated with accidents and improve road safety with better regulation. Analyzing how self-driving cars perform can help developers understand what is working correctly with the autonomous system and which areas need to be improved.

A common approach to get insights about driving performance from driving data is to analyze maneuvers. In fact, more and more authors use this type of analysis in their work [12, 13, 16, 19, 97, 98]. The rationale is that the set and frequency of the maneuvers performed during a trip and the way they are executed can provide relevant information about the driving behavior of the driver during the trip. So far, the driving data normally used in this task is high-frequency telematics (also called automobile sensor data), such as GPS location, velocity and acceleration, and video recordings.

In a previous article [99], we argued that using time-series motifs detection algorithms to extract maneuvers from high-frequency telematics had two main benefits. First, it was more adaptable to small fluctuations in the data than previous methods. And second, it had the advantage of not requiring labels, which is extremely time-consuming to collect. We also noted that analyzing maneuvers through motif detection in telematics data is a promising area of research that is yet to be fully explored.

Recently, Jain et al. [100] proposed a general method to discover motifs in noisy time-series and, in one of their case studies, they concluded that their method was capable of identifying turn maneuvers from automobile sensor data. This work further validates our claim that motifs extracted from driving sensor data are highly related to the actual maneuvers performed.

In this paper, we expand the work done in [99] by proposing TripMD, a complete motif extraction and exploration system that is tailored for the task of analyzing maneuvers and driving behaviors. Other authors have looked into the task of maneuver detection using time-series motifs [100, 101]. However, none of these works propose a full system that extracts motifs from automobile sensor data and summarizes the information in a space-efficient visualization.

Particularly, our main contributions are the following:

- We present TripMD, motif detection and summarization system that was designed to extract relevant driving patterns from a set of trips. It is the first system that not only extracts but also summarizes the main motifs of the provided trips, which allows for an easy investigation of the maneuvers being performed. TripMD expands our previous work by combining the motif extraction step with a motif clustering and summarization step.

- We present a novel representation method called VSAX, which adapts the classical SAX representation from [44] to work with variable-length patterns. This new representation is what allows TripMD to capture maneuvers of variable lengths.

To evaluate the applicability of TripMD to real tasks, we use the UAH-DriveSet naturalistic driving dataset [26] in two experiments. Firstly, we apply our system to the trips performed by a single driver and show that it is capable of extracting a rich set of driving patterns. We also show that these patterns can be used to distinguish between three different driving behaviors of the driver. Secondly, we demonstrate that, using the patterns extracted by TripMD, we are capable of identifying the driving behavior of an unknown driver from a group of drivers whose behavior we know. In other words, the association between driving patterns and driving behavior achieved with TripMD can generalize to unclassified drivers.

The rest of the paper is organized as follows. In Section 5.2, we provide an overview of time-series motifs and motif detection algorithms, which will be helpful to understand TripMD. In Section 5.3, we describe TripMD in detail. Section 5.4 is reserved for two experiments where we showcase our system and demonstrate its usefulness. And Section 5.5 concludes this work and introduces some ideas for future work.

5.2 Preliminaries

In simple terms, a time-series motif is a repeated pattern in the time-series that carries information about the underlying process that generated the time-series. Based on this general definition, there are two main ways of defining how relevant a repeated pattern is, namely based on support or based on similarity [68]. In the support-based definition, the most relevant pattern is the one with the highest number of repetitions, while, in the similarity-based definition, the most relevant pattern is the one with the most identical repetitions. Therefore, the support-based definition extracts more frequent patterns and the support-based definition extracts more similar patterns.

There are two additional constrains that a pattern needs to meet to be considered a time-series motif [65]. Firstly, two subsequences that belong to the same motif cannot overlap in time. This non-overlapping constraint is set to avoid trivial matchings. Secondly, two subsequences need to be at a distance smaller than a predefined radius R to be considered a *match* (and thus to belong to the same pattern).

Note that this second constraint is highly tailored to the use-case. On the one-hand, in most motif detection algorithms, the radius is a parameter that needs to be defined by the user. On the other hand, there are many distances that can be used to compute similarity between subsequences [43, 52] and the user must decide which distance is the most suitable to the specific use-case.

The final constrain appears when the task is not to look for a single motif but to more than one motif. In this case, based on the motif definition, it is possible to order motifs based on their relevance and to extract the top-k most important motifs, which are named the *k-motifs*. However, any two motifs can only coexist in the list of *k-motifs* if their centers (the subsequence that better represents the motif) have a distance higher than $2R$ [65], where R is the radius used to define the motif's matches.

In terms of the distance functions, the most used are the Euclidean distance [53] and the DTW distance [55]. The Euclidean distance performs an one-to-one comparison of single points from the same time location and, because of this, it is very efficient. However, the sequences being compared need to have the same size (or being padded at the end) and the distance is not robust to time-shifts, distortions or differences in phase. On the contrary, the DTW distance is capable of dealing with variable-length sequences and other misalignments by finding an optimal time mapping between the sequences that are being compared. However, this flexibility comes at the cost of efficiency, which is a major concern when analyzing time-series data.

Independently of the distance used, because motif discovery is a task that involves comparing all possible pairs of time-series subsequences, it is very computationally expensive. Thus, a lot of work in motif detection has been focused on making this search more efficient. The most used technique is to reduce the search space by converting the time-series into a low-dimensional representation where the true distance is approximately maintained. Then, we can prune motif candidates in this reduced space and search for the final motifs in the reduce group of candidates. The SAX [44] is the standard example of the this technique. It starts by braking the original time-series into fixed-sized sliding windows and then converting each window into a sequence of letters.

Using the Matrix Profile [69] is another commonly used strategy to speed-up the search for the motifs. The Matrix Profile is a meta time-series that annotates the original time-series by providing the distance and the index of each fixed-size sequence's nearest neighbors, excluding trivial matches. Note that the size of the sequence is the only parameter of the method and the distance used is the Euclidean distance. There are many efficient and fast implementations of the Matrix Profile, either approximate or exact, and after the Matrix Profile is computed, extracting similarity-based motifs is trivial.

When working with telematic data, it is common to have data from several sensors such as the accelerometer and the velocimeter. Even in the case of the accelerometer,

one still has two distinct time-series, namely the lateral and the longitudinal acceleration. Therefore, finding maneuvers in telematic data is a multidimensional problem and as such, we need to apply techniques for detecting multidimensional motifs.

In their work, Tanaka et al. [66] suggested to apply a dimensionality reduction technique in order to reduce the multidimensional time-series into a single dimension, which would simplify the problem back to the one-dimensional case. This is a smart approach as it can easily leverage all the existing motif detection algorithms. However, it has the drawback of information loss. If the time-series data contains relevant information in more than one dimension at the same time (which is our case when using acceleration data), then we won't be able to capture all the relevant motifs with this approach.

Another technique widely used in the multidimensional setting is to apply an one-dimensional motif detection algorithm in each dimension independently and then look for co-occurrences to extract the multidimensional motifs [70, 71, 72, 73]. This setup is much more accurate (since there is no information loss) and it is more flexible (since the search for co-occurrences can be done with an allowance for asynchronous motifs and the rejection of uninformative dimensions). However, this setup is more computationally expensive.

Instead of working on a two step approach, other authors search for the multidimensional motifs directly by concatenating all the dimensions. For instance, Minnen et al. [70] compute the SAX representation for each dimension and concatenate their strings, while in [102] the authors define a new Matrix Profile, the k-dimensional Matrix Profile, that encodes the distance to each sequence's closest neighbor, taking into account k dimensions.

When the goal is to analyze maneuvers, one needs to be able to extract variable-length motifs. In other words, because the same maneuver does not always take the exact same time, it is important to have flexible methods that can extract motifs of different lengths. Even though fixed-length motifs have been the most explored so far, there are some algorithms for the variable-length case.

Most authors propose to apply a fixed-length algorithm in a range of window sizes and then choose the most representative motifs based on their ranking scheme. The work of Nunthanid et al. [74] and Gao and Lin [75] are two examples of this approach. Note, however, that this approach does not work for maneuvers since all the subsequences that belong to a given motif have the same size. If we have a trip with three right-turn maneuvers that have slightly different durations, these algorithms would not be able to identify these three maneuvers as a motif.

Lin's grammar-based method [83] and Tanaka's EMD algorithm [66] take a different approach. They adapt the sequences' representation in the low-dimensional space in order to take into consideration variable-length patterns. However, this adaption leads to algorithms that are not exact, which means that there is no guarantee that the method can find all the variable-length motifs in a given time-series.

5.3 TripMD

TripMD is a system for extracting and analyzing maneuvers from trips performed by a single driver. To achieve this, we needed algorithms that worked on multi-dimensional time-series and, at the same time, were able to extract and analyze variable-length patterns. Having this in mind, our solution has the following two components:

1. A motif extraction algorithm inspired by the algorithm created by Tanaka et al. [66], which was tailored and tuned for the maneuver detection use-case. It includes a discrete variable-length representation (VSAX) based on the widely used SAX [44] and an iterative pattern matching process that extracts motifs in multiple dimensions.

2. A motif clustering and visualization tool based on the Self-Organizing Map model that extracts the most relevant motif patterns and permits the user to quickly analyze them.

In our motif extraction algorithm, we use the support-based definition. In other words, for a certain VSAX pattern, its motif is the largest group of non-overlapping variable-length subsequences with that VSAX representation and in which all the subsequences have a distance lower than a predefined radius R to the motif's center. Because we are working with variable-length motifs, we use the DTW distance [55] to measure similarity between two multi-dimensional subsequences.

In the following subsections, we'll go through each component in more detail.

5.3.1 Motif extraction

The motif extraction step of TripMD receives a list of trip recordings T and some parameters and returns the list M of all the motifs in those trips that meet our support-based definition.

Algorithm 2 presents the motif extraction algorithm. First, we compute the list of sequences V that contains the VSAX representation of T. This step is further described in Section 5.3.1.1, but in short, each element of T is a sequence of symbols that summarize a particular time-series of T.

Next, for each pattern size w, we extract a list of all words in V for that size (W^w). A word in W^w is a sequence of w VSAX symbols that correspond to a variable-length subsequence in T.

Then, for each unique word in W^w, if it exists, we extract its corresponding motif using the algorithm described in Section 5.3.1.2. We add all the motifs found to the list M and the loop ends when no more motifs of a certain pattern size exist.

Algorithm 2: Motif extraction

Input: T: List of multidimensional time-series representing the trips' recordings
$l_{size} \in \mathbb{Z}^+$: Default letter size
$w_{min} \in \mathbb{Z}^+$: Minimum pattern size
$R \in \mathbb{R}^+$: Motif radius
Output: M: List of all motifs.

1 $V \leftarrow$ GetVsaxSequence(T, l_{size})
2 $M \leftarrow \emptyset$
3 $w \leftarrow w_{min}$
4 **repeat**
5 $\quad W^w \leftarrow$ sequence of VSAX words of size w
6 $\quad W^w_{unique} \leftarrow$ list of unique word patterns
7 $\quad$ **forall** $pattern \in W^w_{unique}$ **do**
8 $\quad\quad m \leftarrow$ GetMotif$(pattern, R, W^w, T)$
9 $\quad\quad$ **if** *number of motif members of* $m > 1$ **then**
10 $\quad\quad\quad M \leftarrow M \cup \{m\}$
11 $\quad w \leftarrow w + 1$
12 **until** $|W^w_{unique}| = |W^w|$ *(i.e., no more repeating patterns)*

5.3.1.1 Variable SAX

VSAX is a time-series discretization method that transforms a time-series into a sequence of symbols that captures the general behavior of the original time-series. It serves two main purposes. Firstly, by providing a discretization of the time-series, it allows for a more efficient motif search and, at the same time, reduces the impact of small levels of noise. Secondly, it is capable of splitting the time-series into subsequences of variable lengths depending on the underlying behavior of the time-series, which allows a simple pattern matching algorithm to find variable-length motifs.

Algorithm 3 lists the main steps of the VSAX algorithm. For simplicity, we present the case where the trips are represented as an one-dimensional time-series. The application of the algorithm to the multi-dimensional case is discussed in the last paragraph of the subsection.

To better explain how the algorithm works, we provide a example in Figure 5.1 with a single one-dimensional time-series.

Initially, the time-series is split into fixed-length sliding windows (tss_k). The length of the window is one of the parameters of VSAX, the default letter size. Then, the values in each sliding window are averaged to obtain a discrete value for that window (paa_k), which in turn is converted to a symbol (c_k) based on a predefined segmentation of the time-series domain.

After obtaining the symbol c_k, the pruning phase checks whether tss_k should be concatenated to the previous window tss_{k-1} (line 14 in Algorithm 3). The rationale is that if two consecutive windows have similar behaviors (which translates into being transformed into the same symbol), then they should be a single window and be considered together when searching for motifs. Thus, in the end, we have a sequence

Algorithm 3: Variable SAX (VSAX)

Input: T: List of one-dimensional time-series
$l_{size} \in \mathbb{Z}^+$: Default letter size
Output: V: list of VSAX letter sequences, one sequence per trip
Let α_i, be the symbol that represents the time-series domain $[b_i, b_{i+1}[$, where $1 \leq i \leq 5$.

Function GetVsaxSequence(T, l_{size}):
  $V \leftarrow \emptyset$
  $\{b_i\}_{1 \leq i \leq 6} \leftarrow$ ComputeVsaxBreakpoints(T)
  foreach *time-series ts of* T **do**
    $V_{ts} \leftarrow \emptyset$
    $ts_{size} \leftarrow$ size of the trip's time-series ts
    for $k \leftarrow 1$ **to** $ts_{size} - l_{size}$ **do**
      $tss_k \leftarrow ts[k : k + l_{size}]$
      $paa_k \leftarrow \frac{1}{l_{size}} \sum tss_k[i]$
      $c_k \leftarrow \alpha_i$ s.t. $paa_k \in [b_i, b_{i+1}[$
      if $|V_{ts}| = 0$ *or* $c_k \neq c_{k-1}$ **then**
        $letter_k \leftarrow (c_k, tss_k)$
        $V_{ts} \leftarrow V_{ts} \cup \{letter_k\}$
      else
        $tss_k = tss_{k-1} \cup tss_k$
        $letter_k \leftarrow (c_k, tss_k)$
        $V_{ts} \leftarrow (V_{ts} \setminus \{letter_{k-1}\}) \cup \{letter_k\}$
    $V \leftarrow V \cup \{V_{ts}\}$
  return V

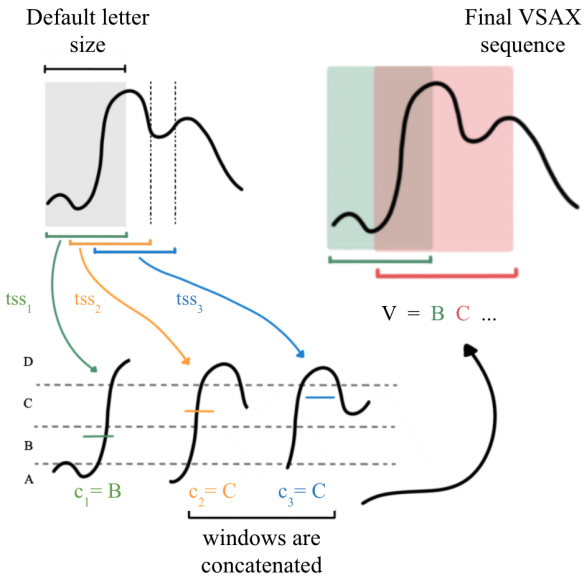


Figure 5.1: Simple example of the Variable SAX representation process.

of symbols V that map to variable-length subsequences of the original time-series and that encode information about the general behavior of the subsequences.

The segmentation of the time-series domain is similar to the way it is done in SAX [44]. Break-points are determined based on the time-series' values and these

break-points define regions in the time-series domain that map to specific symbols. Algorithm 4 details how these break-points are computed in TripMD, assuming the input is an one-dimensional time-series of trips recordings.

Algorithm 4: Variable SAX break-points computation

Input: T: List of one-dimensional time-series
Output: $\{b_i\}_{1 \leq i \leq 6}$: VSAX break-points for T

Function ComputeVsaxBreakpoints(T):
  $joinedT \leftarrow \emptyset$
  foreach *time-series ts of* T **do** $joinedT \leftarrow joinedT \cup ts$
  $b_1 \leftarrow -\infty$
  $b_2 \leftarrow$ Percentile(T, 5)
  $b_3 \leftarrow$ Percentile(T, 15)
  $b_4 \leftarrow$ Percentile(T, 85)
  $b_5 \leftarrow$ Percentile(T, 95)
  $b_6 \leftarrow +\infty$
  return $\{b_1, b_2, b_3, b_4, b_5, b_6\}$

In the original SAX representation [44], break-points are defined so that all regions have equal probability under a Gaussian distribution. However, VSAX uses specific percentiles of the time-series' values to define five regions, namely the 5th, 15th, 85th and 95th percentiles. In general, a driver spends less time performing maneuvers than he does not performing any maneuver and, thus, defining break-points that evenly distribute time-series' values among the regions does not lead to good results in the maneuver detection task. Additionally, since the percentiles are computed over all the trips, any two windows with the same symbol will be guaranteed to be in the same domain region. This is another change compared to SAX, where the break-points are computed independently for each window.

Finally, for multi-dimensional time-series, VSAX can be applied separately to each one-dimensional time-series and then concatenate the resulting symbols in a tuple. For instance, a subsequence of a two-dimensional time-series would be mapped to a tuple of two symbols, one for each dimension. Note however that in the multi-dimensional case, the pruning phase is applied in all the dimensions at the same time. In other words, two consecutive subsequences are only merged if they have the same symbols in all the dimensions. Thus, in this case, each VSAX symbol corresponds to a single variable-length multi-dimensional subsequence of the original time-series.

5.3.1.2 Motif search

The motif search is an iterative process that extracts the motifs of all possible sizes from a VSAX sequence. It was inspired by the motif detection algorithm proposed by Tanaka et al. [66]. At each iteration, it discovers all the motifs with a certain number of VSAX symbols (the pattern size) and then moves to next iteration by increasing the pattern size by one. The minimum pattern size is a parameter of the method and

the iteration stops when no more motifs with the current pattern size can be found. Algorithm 5 lists the main steps to discover the motif associated with a pattern word, if it exists.

Algorithm 5: Motif discovery from a word pattern

Input: $pattern$: pattern word
$R \in \mathbb{R}^+$: Motif radius
W^w: sequence of VSAX words of size w
T: List of multidimensional time-series representing the trips' recordings
Output: $(m_{center}, m_{members})$: center and members of the final motif, it they exist

1 **Function** GetMotif($pattern, R, W^w, T$):
2 $m_{candidates} \leftarrow$ all subsequences with the word $pattern$
3 $D \leftarrow$ matrix of the DTW distances of all subsequence pairs of $m_{candidates}$
4 $count_{max} \leftarrow 0$
5 $m_{mean} \leftarrow \infty$
6 $m_{center} \leftarrow \emptyset$
7 $m_{members} \leftarrow \emptyset$
8 **foreach** *candidate* c *of* $m_{candidates}$ **do**
9 $c_{members} \leftarrow \{s \in m_{candidates} \mid distance(c,s) \leq R \wedge c, s \text{ do not overlap}\}$
10 $count_{new} \leftarrow |c_{members}|$
11 **if** $count_{new} > count_{max}$ **then**
12 $count_{max} \leftarrow count_{new}$
13 $m_{mean} \leftarrow$ Mean distance between $c_{members}$ and c
14 $m_{center} \leftarrow c$
15 $m_{members} \leftarrow c_{members}$
16 **else if** $count_{new} = count_{max}$ **then**
17 $c_{mean} \leftarrow$ Mean distance between $c_{members}$ and c
18 **if** $c_{mean} > m_{mean}$ **then**
19 $count_{max} \leftarrow count_{new}$
20 $m_{mean} \leftarrow c_{mean}$
21 $m_{center} \leftarrow c$
22 $m_{members} \leftarrow c_{members}$
23 **return** $(m_{center}, m_{members})$

To further illustrate the algorithm, Figure 5.2 provides a concrete example. In this case, the pattern word is BC and the motif radius $R = 4$. Initially, we extract all the subsequences with that same pattern to make the pool of the motif's center candidates $c_{candidates}$. In the example, $c_{candidates}$ has 3 subsequences.

From $c_{candidates}$, we compute the matrix of DTW distances for all the candidates (D), which allows us to extract the motif members of a center candidate.

Then, we loop through all subsequences in $c_{candidates}$ and get the corresponding motif members, $m_{candidates}$, assuming that the subsequence c is the center of the motif. Concretely, for a given center candidate c, all the non-overlapping candidates that are within R from the initial candidate are extracted and stored in $c_{members}$ (line 9 of Algorithm 5).

The final motif is defined to be the set of subsequences with the most members ($m_{members}$), and the motif's center is the original candidate that generated that set of members (m_{center}). If no set of more than one member is found, then the motif for that

specific pattern does not exist and both $m_{members}$ and m_{center} are empty. In the example, the final motif exists and has the first subsequence as its center and the remaining two subsequences as its members.

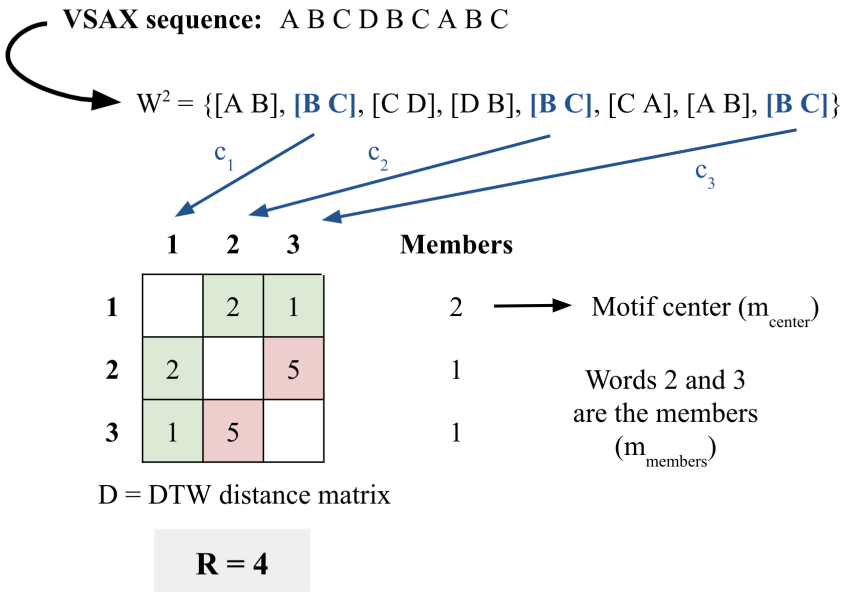


Figure 5.2: Simple example of the motif search process for a single pattern word *BC*.

5.3.2 Motif summarization

In a previous work [103], we proposed a new dimensionality reduction method to summarize and explore the outputs of any motif detection algorithm. The method, called DTW-SOM, is a vanilla Self-Organizing Map [104] with some adaptions to work with time-series motifs. It receives a list of variable-length multi-dimensional motifs and produces a clustering of the motifs' centers and a visualization of the results that is space-efficient.

TripMD leverages the DTW-SOM algorithm to group all the motifs found by the motif search process and to provide a visual summary of the most relevant motifs in the trips under analysis. By summarizing the extracted motifs, the user can quickly analyze the main patterns that are extracted and can better interpret the maneuvers being performed.

However, there is an important step before applying this method. DTW-SOM includes two initialization routines, a random initialization, in which the DTW-SOM network is initialized with a random sample of the motifs, and an anchor initialization, in which the user provides the set of motifs to used for initialization. Since previous experiments indicated that the anchor initialization was more stable than the random initialization [103], TripMD uses the anchor initialization. This means that it has to

include a motif pruning step (Algorithm 6) that computes the most relevant motifs, which are used as the anchors.

Algorithm 6: Motif pruning

Input: M: list of motifs
$R \in \mathbb{R}^+$: Motif radius
Output: P: Pruned list of motifs, ordered by MDL score, from lowest to highest

```
1  scores ← ∅
2  foreach motif m of M do
3  |   mdl_m ← MdlScore(m)
4  |   scores ← scores ∪ {mdl_m}
5  M* ← Sort(M, by = scores)
6  P ← {M*[1]}
7  foreach motif m of M* do
8  |   distances ← ∅
9  |   foreach motif p of P do
10 |   |   d ← DTW distance between the centers of p and m
11 |   |   distances ← distances ∪ {d}
12 |   if d ≤ 2R ∀d ∈ distances then
13 |   |   P ← P ∪ {m}
```

The pruning routine used in TripMD (Algorithm 6) is based on the definition of k-motifs and the non-overlapping requirement discussed in Section 5.2. Given a natural ordering of all the extracted motifs, the k-motif is the highest ranking motif whose center has a distance higher than $2R$ to each of the j-motifs' centers, for $1 \leq j \leq k-1$. Thus, the first pruned motif is the first element of the sorted list of motifs ($M^*[1]$). Then, the next subsequence motif to be added to the list of pruned motifs P is the first motif in the sorted list M^* that has a distance higher than $2R$ to the motif in P. And so forth for the remaining motifs in M^*.

In TripMD (line 5 of Algorithm 6), the ordering is defined by the MDL cost proposed by Tanaka et al. [66]. This score is based on the Minimum Description Length (MDL) principle [76] which states that the *"best model to describe a set of data is the model which minimizes the description length of the entire data set"* [66]. In other words, by using the MDL principle, we are ranking the motifs based on their capacity to compress the original time-series data. In this case, the lower the MDL score, the more relevant the motif is. After pruning, the lowest-scored motifs (P) are used to initialize the DTW-SOM and all the motifs are fed into the algorithm.

TripMD also imposes a constrain on the distance computation of DTW-SOM. The original SOM algorithm [104] uses the Euclidean distance as its distance metric. However, because DTW-SOM needs to work with variable-length subsequences, it uses the DTW distance instead.

As presented in Section 5.2, DTW allows a more flexible comparison of two subsequences by finding the optimal time mapping between them. This flexible mapping is also called time warping. If no constraint is provided, DTW searches are possible

mappings between the two subsequences to find the one with the lowest distance. Alternatively, DTW can limit the maximum time wrapping allowed. In this case, DTW cannot map two time-steps that are farther in time than the maximum warping window. Thus, because the search is constrained, the final mapping will be sub-optimal, and the resulting distance will be higher or equal to the one obtained with an unconstrained version of DTW.

5.3.3 Parameter analysis

So far, TripMD seems to have some parameters that a user needs to set beforehand. VSAX has the default letter size, the motif search has the radius R to define the motifs and the minimum pattern size that initializes the search, and DTW-SOM has the number of training epochs and the maximum warping window. However, TripMD has default values for these parameters, namely:

- **Default letter size:** 1 second
- **Minimum pattern size:** 3 VSAX letters
- **Motif radius:** 0.5th percentile of the distance between all pairs of 3 second subsequences
- **Number of epochs for DTW-SOM:** 20
- **Maximum warping for DTW-SOM:** VSAX's default letter size (or 1 second)

Using the default parameters, the only parameter that the user must provide is the frequency in Hertz of the input time-series, which is trivial. If the user has some particularity in his dataset that makes the default parameters unreasonable, there's always the possibility of overriding the defaults provided by the TripMD.

To better understand how these parameters influence our method, we use a small toy example and test it against varying parameter values. We build a toy example by picking two right turns from a real driver and concatenating them with some random noise. The noise was obtained using an uniform distribution $U(-0.005, 0.005)$. Thus, we obtain the lateral and longitudinal acceleration of a small trip with two right turns connected by random noise.

Since the data has a frequency of 5 Hz, the default settings of TripMD lead to a default letter size of 5 and a minimum pattern size of 3. We set the default motif radius to 0.0684 as this was the value estimated by our method using all the trips of the driver from which we picked the two right turns of the toy example. With these default parameters, we run the motif extraction component of TripMD and obtained two motifs. Figure 5.3 shows one of the motifs extracted, which includes the two subsequences of the original right turns, confirming that the default parameters work well for the toy example.

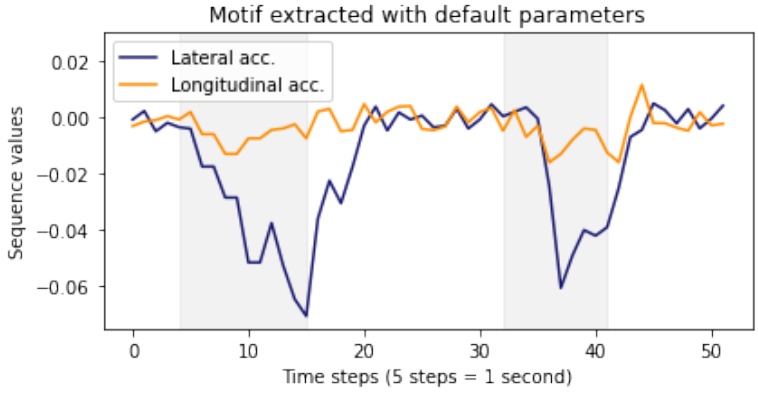


Figure 5.3: Motif extracted from the toy example using the default parameters. It includes the lateral and longitudinal acceleration values of the entire toy example. The two subsequences that belong to the motif are identified by the shaded area.

In TripMD, the default letter size controls the degree of detail used to define driving patterns. This parameter sets the size of the sliding windows used in the VSAX discretization. Thus, if the parameter is too large, the sliding windows will be too large as well, and TripMD will lose the detail necessary to identify relevant maneuvers. On the other hand, if the parameter is too small, the sliding windows will be too short as well, and TripMD will be too sensitive to noise. We define the default value to be one second because in the dataset we had available one second was the right duration to identify simple changes in acceleration that are related to actual maneuvers. However, this should be further tested in other datasets.

Figure 5.4 shows the motifs extracted from the toy example using two default letter sizes, one small (size 2) and one large (size 7). As explained in the previous paragraph, when the window size is small, the motif extracted is also small, and we can never extract the entire turn maneuvers. When the window size is large, the motif extracted expands beyond the subsequences associated with the turns.

The minimum pattern size corresponds to the minimum number of VSAX letters used to build motifs. Setting the default to three letters means that all the motifs will correspond to subsequences with two changes in acceleration. For instance, a simple turn maneuver should include two changes in the lateral acceleration: from zero to absolute high and back again to zero. Three letters are the minimum number of changes to have a meaningful maneuver.

If the minimum pattern size is set to a lower value, TripMD will still extract motifs with more letters. However, the search will take longer to finish. On the other hand, if the minimum pattern size is higher than three, TripMD will not find simple maneuvers such as a right turn or a brake.

As an example, Figure 5.5 shows two motifs extracted with a minimum pattern size 1. Both motifs are parts of the initial motif extracted with the default parameters. Because the motif search algorithms needs to search through these smaller sized motifs,

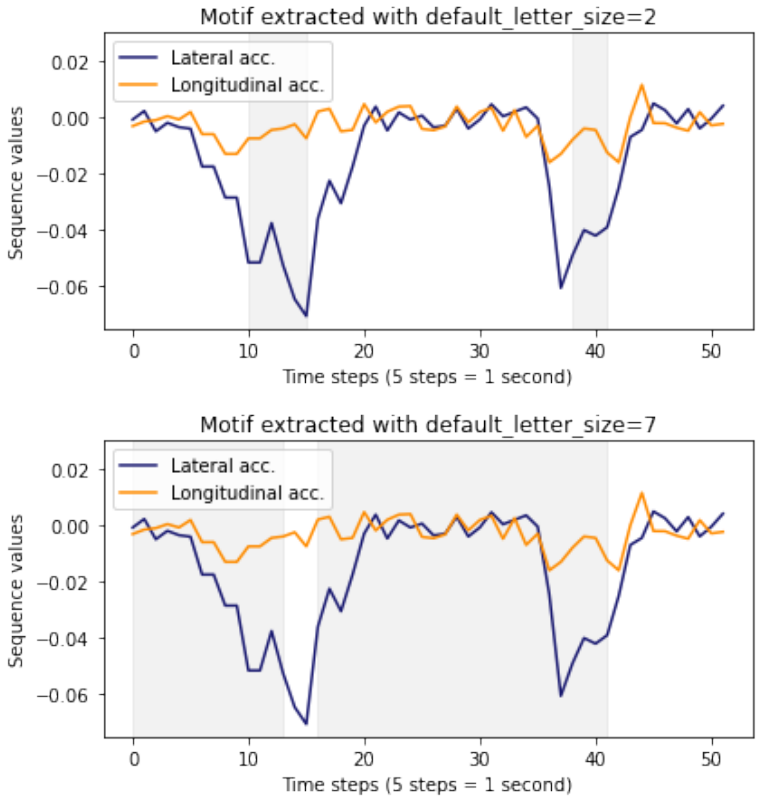


Figure 5.4: Motifs extracted from the toy example with varying default letter sizes. The subsequences that belong to the motifs are identified by the shaded area.

the computation takes longer and does not arrive at any extra meaningful maneuvers.

The motif radius parameter, R, limits the subsequences that are considered to be a motif. Even if two subsequences have the same pattern word, they will only belong to the same motif if their DTW distance is lower than R. Thus, if R is too small, very few motifs will be found. And if R is too large, clearly different subsequences will be added to the same motif, which will result in poor results.

When estimating this parameter, we assume that motifs are rare and that most pairs of subsequences in a trip will not be the same maneuver. Thus, we use a small percentile on the distribution of the distance between random pairs of subsequences to guarantee that we use a R large enough to find motifs without the risk of grouping different maneuvers in the same motif.

Going back to the toy example, if the radius is set to value lower than 0.04, no motifs are extracted. However, setting R higher or equal to 0.05 allows us to extract the motif with the two turn maneuvers. The difference, however, between using a radius of 0.05 and a radius of 0.07 is the extra patterns that are extracted when one increases R. As an example, Figure 5.6 shows a motif extracted with a motif radius of 0.1.

The final two parameters are related to the DTW-SOM method. The number of

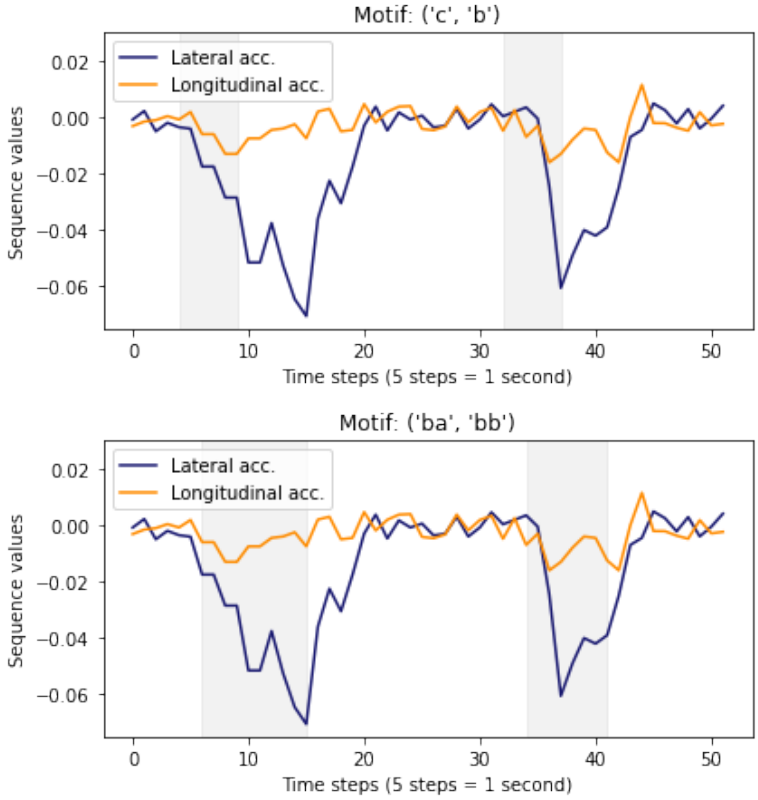


Figure 5.5: Motifs extracted from the toy example using a minimum pattern size 1. One of the motifs corresponds to a pattern size of 1 and the other corresponds to a pattern size of 2. The subsequences that belong to the motifs are identified by the shaded area.

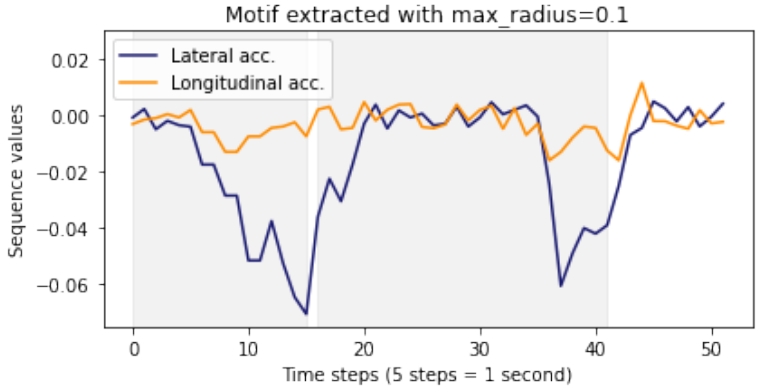


Figure 5.6: Motif extracted from the toy example using a motif radius of 0.1. The subsequences that belong to the motif are identified by the shaded area.

epochs sets the number of training iterations for the SOM. The higher the number of epochs, the longer the model will train (and the longer the full run of TripMD will be). However, if the number of epochs is too small, the final summary of the motifs will be poor. Since we are using the anchor initialization, we don't need as many epochs for the SOM training to converge. From our experiments, the default of 20 epochs is

sufficient for a proper summarization.

As for the maximum warping, it limits the time-warping allowed in the DTW distance computation. If we decrease this parameter, the maximum warping window is small, and the distance between the two subsequences will be closer to their Euclidean distance. Alternatively, if we allow for more warping, the clustering of the motifs will be more flexible and allow more misaligned sequences to be grouped in the same SOM unit.

There are two reasons for limiting the warping window. Firstly, since we are reducing the search for an optimal mapping, the computation is faster. Secondly, if the warping window is not constrained, motifs with a higher degree of time misalignment will be grouped in the same SOM unit, and the final motif summarization will be distorted.

5.4 Evaluation and discussion

To evaluate TripMD, we use the UAH-DriveSet [26], a publicly available naturalistic driving dataset including recorded trips from six different drivers that traveled in two specific routes in Madrid, Spain. The authors asked the volunteers to drive in these two routes mimicking three different driving behaviors - normal, aggressive and drowsy. Using their DriveSafe app [27, 28], the authors collected raw data from the accelerometer, GPS and camera of a smartphone mounted in the car and processed these signals to enrich the final dataset.

In the first experiment, we pick a single driver and explore in detail the outputs obtained from TripMD. Particularly, we do an exploratory analysis of the motifs extracted by TripMD and showcase the visualizations provided by our system.

In the second experiment, we focus on the task of identifying driving behaviors. We apply TripMD to the entire UAH-DriveSet. Then, using the known driving behaviors of all but one driver, we assign behavior scores to each motif cluster. Finally, we use those cluster behavior scores and the motifs extracted from the left-out driver to predict the behavior of each of that driver's trip.

In both experiments, we use the two-dimensional time-series of the lateral and longitudinal acceleration recordings. The recordings are already aligned with the correct car axis and denoised with a Kalman filter, which means we can use them directly. The data has a frequency of 10Hz, however, in order to speed computation and further reduce noise, we down-sample the time-series to a 5HZ frequency. Additionally, we use all the default parameters for TripMD as we found that they work well for this dataset.

The code to reproduce all the experiments can be consulted in our repository [1].

[1] https://github.com/misilva73/tripMD

5.4.1 Analyzing a single driver with TripMD

To showcase how our system can be used to explore the driving behavior of a single person, we run TripMD on the seven trips performed by one of the drivers in the UAH-DriveSet. This driver completed four trips in the secondary road route (two normal, one aggressive and one drowsy) and three trips in the motorway route (one for each driving behavior).

The motif detection component found 281 motifs and, from these, 17 motifs were used to initialized the DTW-SOM model. These 17 motifs were the result of the pruning step presented in Section 5.3. The DTW-SOM model was initialized with a 5X5 grid since it is the smallest square grid that can contain the 17 pruned motifs. Then the 281 motifs were assigned to each DTW-SOM unit in the grid. DTW-SOM builds an optimal assignment by reducing the DTW distance between the units and their motif's centers. DTW-SOM also provides a visualization of the clusters in a two-dimensional grid (or network) that conserves the local similarity of the data. This means that two neighboring clusters in the two-dimensional network are similar.

Figure 5.7 shows the first visualizations provided by TripMD for the driver. It contains the lateral and longitudinal acceleration of each DTW-SOM unit, placed in the two-dimensional network. A unit here is a multi-dimensional subsequence that represents the cluster in a particular part of the DTW-SOM grid and thus this plot provides a summary of the main driving patterns extracted from the driver.

From this first chart, we can already see that TripMD is able of identifying a rich set of driving patterns, with lengths ranging from 1.5 to 3 seconds. It includes simple maneuvers, such as unit 22 that relates with a simple left turn without changes in longitudinal acceleration, and more complex maneuvers, for instance, unit 0 that corresponds to a right turn with acceleration.

Additionally, the grid maintains some local similarity, with adjacent units showing more similar acceleration patterns than units that are not adjacent. As an example, the neighboring units 15, 16, 20, and 21 all have similar driving patterns, with a clear brake maneuver and a slightly positive lateral acceleration.

Figure 5.8 contains two additional visualizations provided by TripMD for the driver. These plots are classical ways of visualizing a SOM network and represent different information about each of the clusters arranged in the two-dimensional network. In both charts, the arrangement of the units in the two-dimensional grid is consistent to Figure 5.7.

The first chart is called U-Matrix and it shows how similar each unit is to its direct neighbors in the two-dimensional network, where the brighter the color, the closer a unit is to its neighbor. This visualization is helpful to understand where are the major groups of clusters within the network. For instance, the upper-right corner has a clearly defined groups of four cluster that are very similar, which corresponds to the units 15, 16, 20 and 21 with the brake maneuver discussed above.

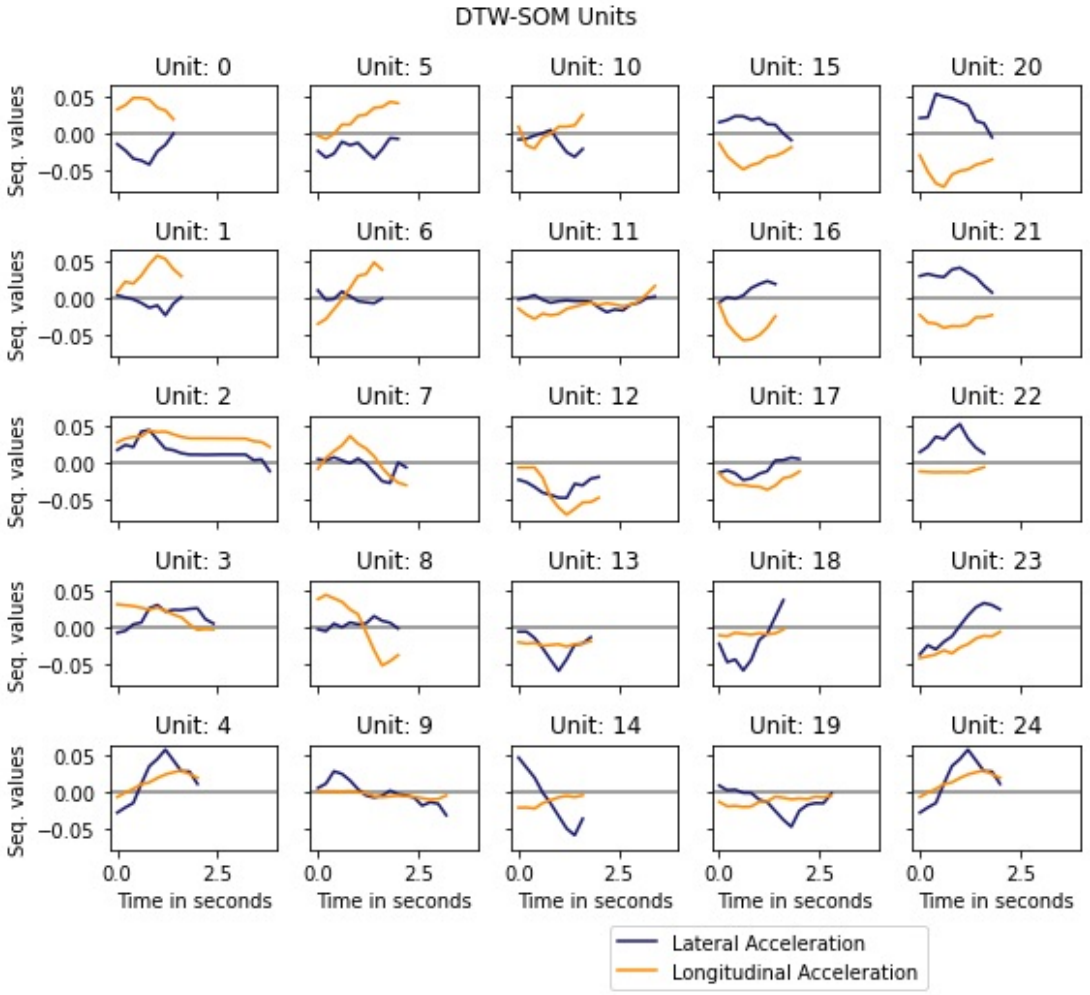


Figure 5.7: Lateral and longitudinal acceleration of the DTW-SOM's units. They were obtained by applying TripMD to the trips of a single driver from the UAH-DriveSet. Units are placed in the DTW-SOM two-dimensional grid.

The second chart is called Winner Matrix and it provides information about the cluster size of each unit. Particularly, it displays the exact number of motifs in each cluster on top of corresponding unit. This plot can be used to gauge how relevant each driving pattern is. For instance, unit 24 in the lower right corner contains no motifs, which means that this pattern is not needed to summarize the driver's behavior.

Besides these default TripMD plots, Figure 5.9 contains information about the distribution of each driving behavior in the clusters extracted by our system. For each driving behavior, we compute the number of motif subsequences from the trips with that behavior that belong to each DTW-SOM cluster. Then, we divide each cluster count by the total number of motif subsequences for all the driver's trips that belong to that cluster to achieve the rate presented in the plots. So, for instance, 80% of the motif subsequences associated to motifs that belong to the cluster 4 come from trips with a drowsy behavior.

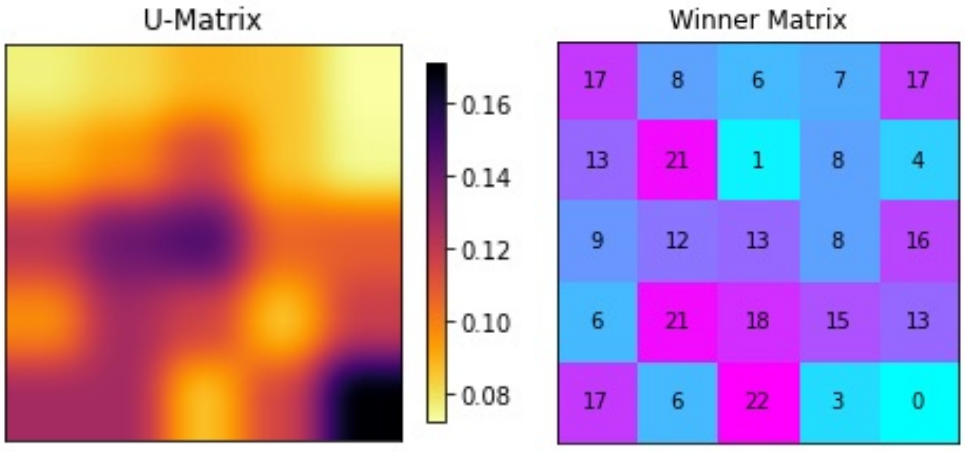


Figure 5.8: U-matrix and Winner Matrix of the DTW-SOM. The model was trained on the motifs extracted from the trips of a single driver from the UAH-DriveSet.

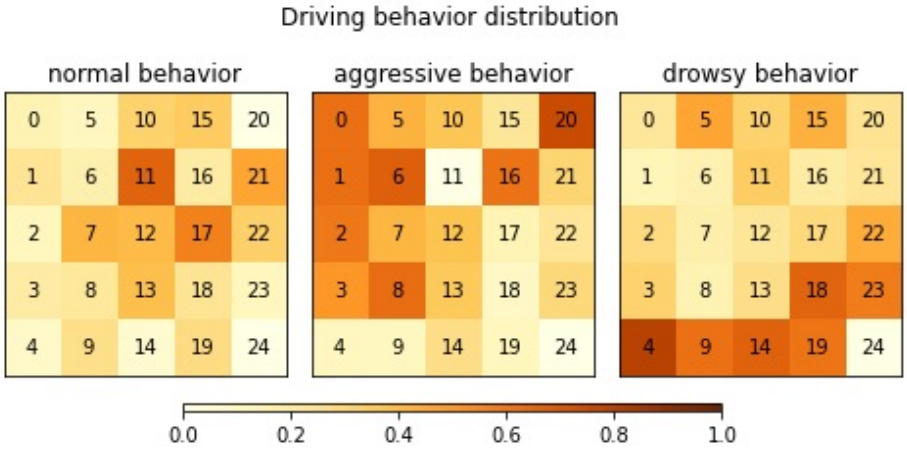


Figure 5.9: Driving behavior rates for each DTW-SOM unit. For each unit, it shows the percentage of motif subsequences that come from each driving behavior.

Interestingly, we can see that the three driving behaviors have very different distributions of their motif subsequences among the clusters. Clusters 11 and 17 have a clear majority of subsequences from normal trips and these clusters relate to a "no maneuver"pattern and a soft brake, respectively.

The aggressive trips cover a higher variety of patterns, with 7 clusters showing a clear majority of subsequences from these trips. This increase in representation is expected as more motif subsequences will be extracted from trips where the driver performs more maneuvers. Most of these 7 clusters contain sharp acceleration patterns, which is usually associated with aggressive driving. Examples are the right turn with a pronounced brake in unit 20, the brake-acceleration pattern in unit 6 and the quick acceleration in unit 1. These sharp acceleration maneuvers without lateral movements are specially telling as they are associated with tailgating behavior, which in turn is a classical aggressive driving behavior.

Finally, the clusters with higher rates of subsequences coming from drowsy trips are located in the lower row of the DTW-SOM grid, excluding unit 24. Units 14 and 18 contain a drift pattern, which is made of two consecutive lateral movements in opposing sides. It is very interesting to see these patterns here as they are usually present in cases where a tired driver lets the car deviate from a lane and then quickly recovers with a sharp turn.

5.4.2 Identifying driving behavior with TripMD

From the first experiment, we see that TripMD can summarize the trips from a single driver so that different driving behaviors can be identified. However, to further test our system, we focus on a harder task, namely, identifying the driving behavior of an unknown driver from a set of drivers whose behavior we know.

To accomplish this, we apply TripMD to the entire UAH-DriveSet and retrieve the main driving patterns of all those trips. Then, using the known driving behavior of five drivers (the training drivers), we derive scores for all the trips of the remaining driver (the testing driver). This testing driver was the same used in the first experiment. For each trip of the testing driver, a single score is computed for each of the three behaviors - normal, aggressive and drowsy. To compute the score for a specific testing trip and a given behavior b, we use the following process:

1. For each DWT-SOM cluster c_i:
   a) Compute the rate $r_i^b = \frac{n_i^b}{n_i}$, where n_i^b is the number of subsequences in cluster c_i that belong to training trips of the behavior b and n_i is the total number of subsequences in cluster c_i that belong to training trips.
   b) Compute $\hat{n_i}$, which is the number of motif subsequences in cluster c_i that belong to testing trip.
   c) Derive the behavior score of cluster c_i as $s_i^b = r_i^b \times \hat{n_i}$.
2. Compute the trip's behavior score as $s^b = \sum_{i=1}^{k} s_i^b$, where k is the number of DWT-SOM clusters.

After computing the three behavior scores for a testing trip, its predicted behavior is simply the behavior with the highest score. Finally, we compare the predicted behavior of each testing driver's trips with the real behavior performed in those trips. Table 5.1 summarizes the results.

The testing driver contains seven trips and, from these, TripMD assigns the correct behavior to all but one trip. The trip where the predicted behavior does not match the real behavior is the drowsy trip of the secondary road.

To further illustrate these results, Figure 5.10 shows the behavior rates of the DTW-SOM clusters for the training drivers and the testing driver. Here we can observe that

Table 5.1: Behavior scores for each testing trip. The score of the predicted behavior for each trip is highlighted in bold. The column named *Behavior* contains the real behavior of the trip.

Route	Behavior	Aggressive score	Drowsy score	Normal score
Motorway	Normal	94.1	78.3	**95.5**
Secondary	Normal	35.5	41.1	**43.3**
Secondary	Normal	47.0	35.9	**48.1**
Motorway	Aggressive	**103.1**	72.6	96.3
Secondary	Aggressive	**91.0**	52.2	77.8
Motorway	Drowsy	97.5	**137.9**	120.6
Secondary	Drowsy	69.7	81.9	**82.4**

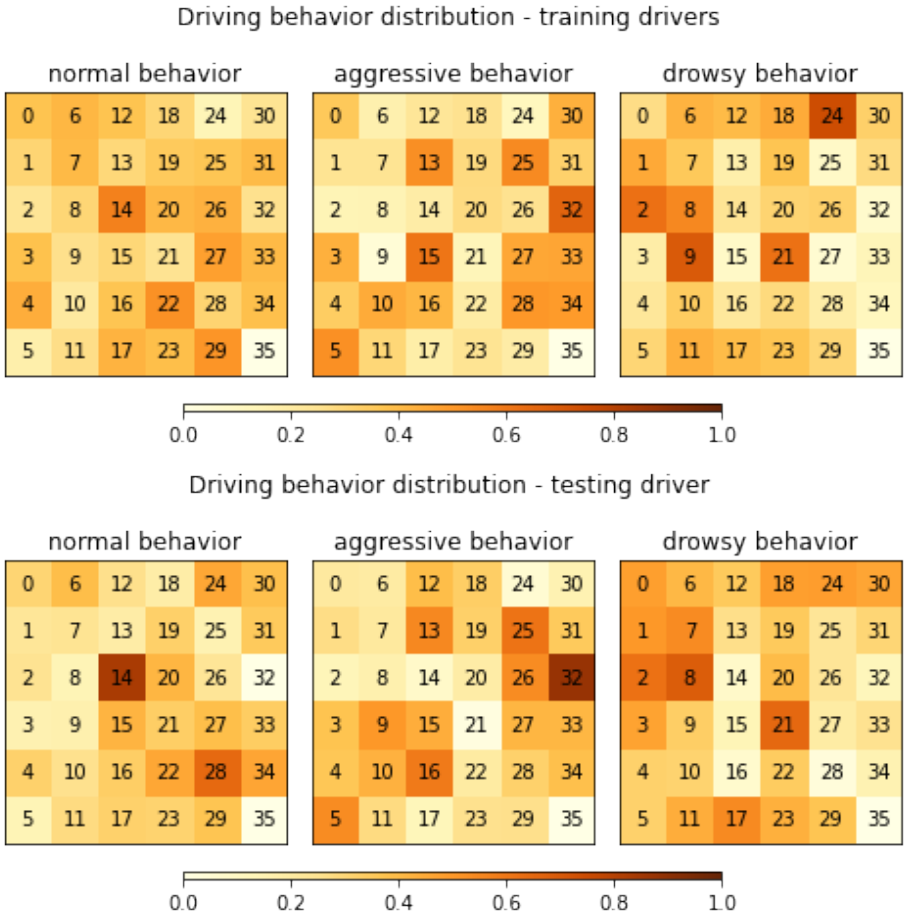


Figure 5.10: Behavior rates for the DTW-SOM units. For each unit, it shows the percentage of motif subsequences that come from each driving behavior. The three plots at the top contain the rates of the training trips while the three plots at the bottom contain the rates of the testing trips.

the behavior distributions of the training drivers are close to behavior distributions of the testing driver. For instance, units 14 has a high normal behavior rate in both sets

of trips, units 13, 25 and 32 have similarly high aggressive behavior rates and units 2, 8 and 21 have high drowsy behavior rates for both the testing and training drivers.

Going back to the scores in Table 5.1, we note that the trips with the normal behavior and the drowsy trip on the secondary road route have two predicted behaviors with very close scores, which may indicate that TripMD is not consistently catching these particular behaviors well.

To further explore this issue, we use a resampling method to measure the stability of the computed scores. Concretely, we apply random sampling with replacement to the list of motifs extracted by TripMD and update the DTW-SOM clusters. The DTW-SOM is not retrained, and thus its units do not change. Instead, only the group of motifs assigned to each cluster are sampled. Since we take a sample of the same size as the original list of motifs, some motifs are repeated multiple times while others are never sampled.

With the new sampled motifs, we then apply the same procedure to compute the behavior score of all the trips performed by the testing driver. This sampling method can be repeated many times and, for each sample, we get a new estimation of each trip's behavior scores. Figure 5.11 shows the average behavior scores and their standard deviations for the testing driver's trips obtained after 1000 samples.

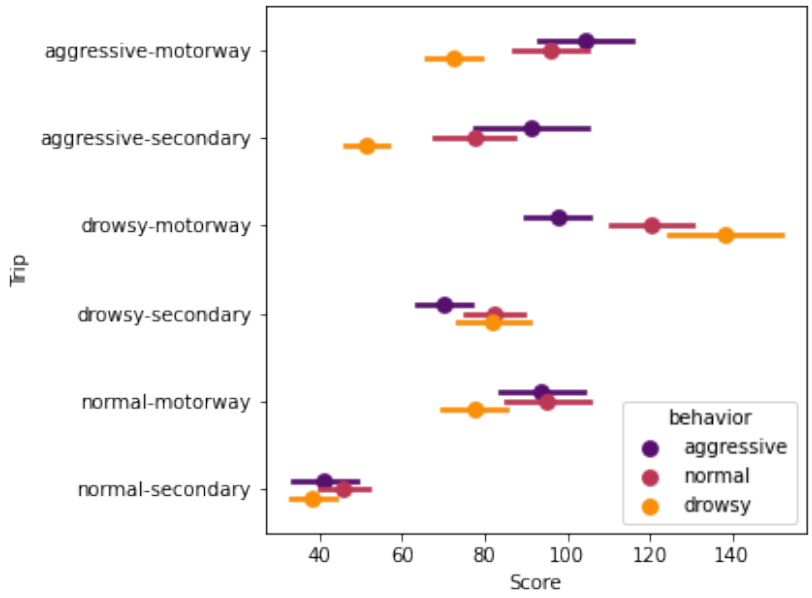


Figure 5.11: Summary statistics of the each testing trip's behavior scores obtained with 1000 samples with replacement. The points encode the average scores while the errors bars encode the standard deviation.

As expected, in the drowsy trip of the secondary road, the distribution of the scores for the normal and drowsy behavior are very close. Similarly, some scores for the normal trips in both the secondary and motorway routes overlap, which agrees with the initial assessment that TripMD is not strong at identifying the trips with the

normal behaviors and the drowsy trip on the secondary road.

On the other hand, the aggressive trips and the drowsy trip on the motorway have the correct behavior score more consistently higher than the remaining behavior scores. Thus, in these cases, TripMD is identifying successfully the underlying behavior of the driver.

5.5 Conclusion

In this paper, we propose a system called TripMD, which identifies the main driving patterns from sensor recordings such as acceleration and velocity. Compared to previous work, our system not only extracts the time-series patterns present in trips recordings but is also capable of summarizing those patterns in a space-efficient visualization. This feature is highlighted in our first experiment, where we demonstrate that TripMD can discover a wide range of driving patterns from the trips performed by a single driver of the UAH-DriveSet dataset. We also conclude that the three driving behaviors marked in the dataset (normal, aggressive and drowsy) have distinct distributions among the extracted driving patterns, which can be used to determine the driving behavior of a driver from the behaviors of other drivers.

Even though the results seem promising, there are still areas of improvement. Firstly, because of the Variable SAX discretization, the motif detection algorithm used in TripMD is not an *exact* algorithm. This means that we cannot guarantee to extract all the variable-length motifs. There are some new motif detection algorithms that claim to be exact, however, we could not find one that was capable of extracting motifs with member's subsequences of difference sizes. Thus, investigating exact motif detection algorithms that work with variable-length motifs could be an interesting line for improvement.

Additionally, we should further test TripMD with more datasets and different tasks, such as understanding whether TripMD can be used to distinguish between drivers with prior accidents from drivers without accidents and to inform car insurance pricing models. Testing TripMD with different datasets would also be helpful to further validate and fine-tune the default parameters of the system such as the default letter size. Another part that could be further tested with other datasets is the choice of percentiles for the VSAX representation.

Chapter 6

Conclusion

6.1 Goals and motivation

In the last two decades, telematics' technology and usage have steadily increased, with applications ranging from early detection of car malfunctions to fuel consumption optimization becoming ever more popularized.

Car insurance, in particular, is one of the fields that has been most impacted by telematics, after the introduction of UBI policies. These insurance schemes use driving data to assess the driving of each client and to provide a more accurate pricing. Practitioners agree that UBI can be beneficial for insurers and customers. However, these benefits can only materialize if insurers can measure driving behavior and performance accurately and reliably. Therefore, in this work, we explore how we can leverage telematics data to provide insights into driving behavior.

It is important to note that identifying driving behavior from telematics data is relevant for other applications besides insurance. For instance, car manufacturers can use this technology to uncover which factors are associated with accidents and improve their car's safety systems. Policymakers can use the same strategy to improve regulation and make roads safer. Large transportation fleets can also use telematics to understand how driving behavior affects fuel consumption and optimize it.

When analyzing driving behavior, a common approach is to analyze maneuvers. Maneuvers are the logical blocks of driving and, thus, identifying maneuvers and determining how drivers perform them provides a unique perspective into their driving behavior. However, extracting maneuvers from high-frequency telematics data such as acceleration and velocity is not straightforward.

Current literature on this topic can be broken down into two diverging methodologies, namely, the fixed thresholds strategy[12, 13, 14, 15], and the rolling windows strategy[16, 17, 18, 19, 20, 21, 22, 23, 24].

As the name suggests, the fixed thresholds strategy involves using thresholds on acceleration and/or other inertial measurements to define the start and the end of each specific maneuver. This strategy has the benefit of being simple, easy to interpret, and very computationally efficient. Yet, it requires fine-tuning and expert judgment to implement in a given application. Added to this, small changes in the specific thresholds or the underlying data can lead to huge impacts to the final results and, hence, the methods based on the fixed thresholds strategy are not very adaptable and require a lot of "manual"work to implement.

On the other hand, the rolling windows strategy takes an entirely different approach. First, trips are broken into fixed-sized time windows that can have some level of overlap. Secondly, a detection method is used in each window to identify each maneuver is being performed (if any). Here, the window size and the level of overlap need to be set beforehand and these parameters can have a massive impact on the final results. The window size, in particular, is known to heavily influence how well maneuver can be detected and different maneuvers may require different window sizes.

Consequently, a disadvantage of this type of method is the inflexibility of the window size and overlap parameters.

Another issue with the rolling windows strategy is the need for labels. It is known from the time-series mining community that using unsupervised learning approaches in following windows leads to meaningless results Keogh and Lin [25]. Thus, we either use rule-based detection, which inherits the problems of the fixed-threshold strategy, or we used supervised learning models, which require labeled data and hence a lot of manual data collection work.

For these reasons, there was a clear opportunity to investigate more flexible methods that can detect maneuvers in high-frequency telematics data without the need for large datasets of labeled maneuvers. Interestingly, the community looking at this problem was not yet leveraging the fact that detecting maneuvers from telematics data is essentially a time-series mining task and, as such, taking advantage of the large body of research already done by the time-series mining community.

6.2 Work overview

Our work aimed to answer three specific questions. Firstly, we wished to explore time-series data mining techniques and tailor them to the task of detecting maneuvers from acceleration data. In other words, we wanted to validate whether we could leverage tools from the time-series data mining community to build an adaptable method capable of detecting maneuvers from acceleration data without the need for labels.

Thus, the initial part of our work was dedicated to answering this question (Chapter 3), where we looked into the use of time-series motif detection. In short, time-series motifs are over-represented subsequences in a time series [65]. We hypothesized that over-represented segments of time-series built from acceleration recordings were highly connected to maneuvers and thus we aimed to explore the relationship between the most relevant motifs of a trip and the maneuvers performed during that trip.

To achieve this, we implemented a slightly modified version of a well-known motif detection algorithm (the EMD algorithm by Tanaka et al. [66]) and tested it in a naturalistic driving dataset with trip recordings from six different drivers and two specific routes in Madrid, Spain (the UAH-DriveSet by Romera et al. [26]). Firstly, we tried to identify acceleration and braking maneuvers from the time-series of longitudinal acceleration, Secondly, we applied the same approach to the lateral acceleration time-series and sought to detect turning maneuvers.

After a systematic exploration of the resulting motifs, we found that the EMD algorithm was capable of extracting both simple and more complex maneuvers, thus proving that motif discovery can be used as a flexible strategy to detect maneuvers in telematics data without the use of labeled data.

We also concluded that the number of motifs extracted from the trip recordings was so large that doing a manual exploration was unfeasible. This led us to our second

research question, namely, how we could summarize the motifs extracted (which in the case of our use-case represent maneuvers) in a space-efficient visualization?

To address this question, we proposed DTW-SOM (Section 5), a new version of the SOM (a well-known feature reduction and visualization algorithm). In short, DTW-SOM is a vanilla Self-Organizing Map with three main differences, namely (1) the use of the Dynamic Time Warping distance instead of the Euclidean distance, (2) the adoption of two new network initialization routines (a random sample initialization and an anchor initialization) and (3) the adjustment of the *Adaptation* phase of the training to work with variable-length time-series sequences.

To test DTW-SOM, we used a synthetic dataset and two real time-series datasets from the UCR Time Series Classification Archive [94]. In other words, we used DTW-SOM in three different datasets, not related to telematics or maneuvers. Therefore, even though we create DTW-SOM with the maneuver detection task in mind, this method is more general and can be helpful to explore motifs extracted from many different domains. In all the datasets tested, we found DTW-SOM to be capable of extracting relevant information from a set of motifs and display it in a visualization that is space-efficient.

Finally, the last part of the work was dedicated to improving motif detection and summarization in order to answer the third research question, i.e, whether we could use the maneuvers extracted from telematics data to get insights into the driving behavior of a single driver. We had some requirements for the final system, namely:

- Detecting variable-length motifs.
- Working with multi-dimensional time-series.
- Providing a segmentation of the motifs into groups meaningful for the maneuver detection and analysis tasks.

The result was a new method called TripMD (Chapter 5). This method is a motif extraction and exploration system, tailored for the task of analyzing maneuvers and driving behaviors. TripMD not only extracts but also summarizes the main motifs of the provided trips and allows for an easy investigation of the maneuvers being performed.

It is important to note that, besides combining and adapting previous methods in a new system fitted for the task at hand, TripMD includes two novel pieces of work. The first is the motif summarization method from Chapter 4, DTW-SOM. DTW-SOM is what allows TripMD to group and summarize the motifs extracted from the trips. The second is VSAX (Chapter 5), a new representation method which adapts the classical SAX representation from [44] to work with variable-length patterns. This new representation is what allows TripMD to capture maneuvers of variable lengths.

Using the same naturalist driving dataset, the UAH-DriveSet, we did two distinct experiments (Chapter 5) to assess whether the outputs produced by TripMD could be used to extract insights into driving behavior.

In the first experiment, we focused on a single driver and explored the results obtained by processing all the trips produced by that driver. Since each trip has a specific behavior associated with it (i.e., normal, aggressive, and drowsy), we were able to map those behaviors to the maneuvers extracted with TripMD.

There, we noticed that TripMD identified a rich set of driving patterns, ranging from 1.5 to 3 seconds. The outputs included simple maneuvers (e.g., left turns with constant forward acceleration) and more complex maneuvers (e.g. forward accelerating turns).

More interestingly, we also observed that the three driving behaviors present in the data had very distinct distributions among the extracted driving patterns. For instance, the normal trips had a majority of presence in the motifs associated with soft maneuvers such as the "no maneuver"pattern and the soft brake.

On the other hand, the subsequences from aggressive trips showed a large representation on the motifs linked to sharp acceleration patterns, for example, right turns with a pronounced brake or the brake-acceleration maneuver (i.e. a rapid brake followed by a sharp acceleration). These sharp acceleration maneuvers are especially telling as they are associated with tailgating behavior, which in turn is a known aggressive driving behavior.

As for the drowsy trips, they were more present in the motifs showing a drift pattern, which is made of two consecutive lateral movements in opposing sides. It was very interesting to see these patterns here as they are usually present in cases where a tired driver lets the car deviate from a lane and then quickly recovers with a sharp turn.

Therefore, we were able to conclude from our first experiment that TripMD is capable of summarizing the trips from a single driver so that different driving behaviors can be identified. Next, to further evaluate TripMD, we focused on a more difficult task, i.e, identifying the driving behavior of an unknown driver from a set of known drivers.

To achieve this, we applied TripMD to the entire UAH-DriveSet and extracted its driving patterns. Then, we used the patterns of five of the six drivers to compute scores of each of the three driving behaviors (normal, aggressive, and drowsy). Then, we predicted the behavior of each trip of the remaining driver by comparing the distribution of the trips among the motifs and their corresponding behavior scores.

After comparing the predicted behavior with the real behavior of each trip, we noticed that TriMD was successfully identifying the driving behavior of the aggressive trips and one of the drowsy trips. On the other hand, in the normal trips and the other drowsy trip, TripMD is not as consistent at identifying the underlying behavior.

These results are very promising since they indicate that motifs extracted from time-series of telematics recordings do hold meaningful information for the task of identifying driving behavior. Even though the results are positive, there are still areas of improvement, which we will explore in the next section.

6.3 Limitations and future work

One of the main limitations of this work relates to the lack of public datasets. It is extremely rare to find publicly available data with high-frequency telematics recordings plus information about driving behavior. As such, for the entire work, we only used a single dataset to test. Without a doubt, using a single dataset led to some bias in our conclusions.

Therefore, one possible approach to further develop this work would be to apply TripMD to other datasets. On one hand, testing TripMD with different datasets would further validate and fine-tune the default parameters of the system.

On the other hand, different datasets would allow us to better understand how robust TripMD is at identifying driving behaviors. In our analysis, TripMD was very reliable at identifying the aggressive trips and not so reliable for the normal and drowsy trips. However, due to the nature of the analysis, the actual trips included in the data are a major factor in the conclusions. Thus, varying the data would provide more data points and, thus, an more accurate view of the performance of our system.

Another limitation of this work, and, in particular, TripMD, is the low scalability. This is an area that we did not focus on because we only wanted to prove the validity of motif detection for analyzing driving behavior. Nevertheless, when running TripMD on the UAH-dataset, we noticed that the motif detection component of the system would take a lot of time and computational resources. This means that TripMD would not be feasible for very large datasets.

Motif detection is not a computationally light task. Essentially, it requires a comparison of all possible pairs of subsequences and, when we add the option of variable length subsequences, the number of possible pairs of subsequences explodes. A lot of effort has been put into designing efficient motif detection algorithms. However, none of the more efficient algorithms we found in the literature were suited for the task of maneuver detection. In this setup, another line of research would be to either improve the efficiency of TripMD or start from another motif detection algorithm (that is already more efficient) and adapt it to the task of maneuver detection.

Another shortcoming of TripMD is that it is not an *exact* algorithm. In order words, because of the Variable SAX discretization, we cannot guarantee to extract all the variable-length motifs. Some new motif detection algorithms claim to be exact, however, we could not find one that was capable of extracting motifs with truly variable lengths. Thus, investigating exact motif detection algorithms that work with variable-length motifs could be an interesting line for improvement.

Lastly, even though this would not address a particular limitation of our work, an interesting area of future research would be to utilize TripMD, and motif detection more generally, in different tasks. As an example, one could investigate whether motif detection can be used to distinguish between drivers with prior accidents and drivers without accidents, which in turn could be used to inform pricing models for car insurance.

Bibliography

[1] S. Nora and A. Minc. *L'Informatisation de la Societé*. fr. Paris: La Documentation française, 1978. ISBN: 2-02-004974-0.

[2] B. C. Burrows. "Book Reviews : Simon MORA and Alain MINC, The computerisation of society. A report to the President of France (MIT Press, Massachusetts and London, 1980). ISBN 0-262-14031-4. Cabinet Office, Advisory Council for Applied Re search and Development. Information Technology (HMSO, London, 1980), ISBN 0-11-6308184." In: *Journal of Information Science* 3.1 (1981). _eprint: https://doi.org/10.1177/016555158100300110, pp. 49–51. DOI: 10.1177/016555158100300110. URL: https://doi.org/10.1177/016555158100300110.

[3] S. Husnjak, D. Peraković, I. Forenbacher, and M. Mumdziev. "Telematics System in Usage Based Motor Insurance." en. In: *Procedia Engineering*. 25th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2014 100 (Jan. 2015), pp. 816–825. ISSN: 1877-7058. DOI: 10.1016/j.proeng.2015.01.436. URL: http://www.sciencedirect.com/science/article/pii/S1877705815004634 (visited on 07/13/2020).

[4] J. Wahlström, I. Skog, and P. Händel. "Smartphone-Based Vehicle Telematics: A Ten-Year Anniversary." In: *IEEE Transactions on Intelligent Transportation Systems* 18.10 (Oct. 2017). Conference Name: IEEE Transactions on Intelligent Transportation Systems, pp. 2802–2825. ISSN: 1558-0016. DOI: 10.1109/TITS.2017.2680468.

[5] S. Iwan, K. Małecki, and D. Stalmach. "Utilization of Mobile Applications for the Improvement of Traffic Management Systems." en. In: *Telematics - Support for Transport*. Ed. by J. Mikulski. Communications in Computer and Information Science. Berlin, Heidelberg: Springer, 2014, pp. 48–58. ISBN: 978-3-662-45317-9. DOI: 10.1007/978-3-662-45317-9_6.

[6] J. Mikulski. "Using Telematics in Transport." en. In: *Transport Systems Telematics*. Ed. by J. Mikulski. Communications in Computer and Information Science. Berlin, Heidelberg: Springer, 2010, pp. 175–182. ISBN: 978-3-642-16472-9. DOI: 10.1007/978-3-642-16472-9_19.

[7] S. A. Shaheen and A. P. Cohen. "Carsharing and Personal Vehicle Services: Worldwide Market Developments and Emerging Trends." In: *International Journal of Sustainable Transportation* 7.1 (Jan. 2013). Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/15568318.2012.660103, pp. 5–34. ISSN: 1556-8318. DOI: 10.1080/15568318.2012.660103. URL: https://doi.org/10.1080/15568318.2012.660103 (visited on 07/27/2020).

[8] A. Fotouhi, R. Yusof, R. Rahmani, S. Mekhilef, and N. Shateri. "A review on the applications of driving data and traffic information for vehicles' energy conservation." en. In: *Renewable and Sustainable Energy Reviews* 37 (Sept. 2014), pp. 822–833. ISSN: 1364-0321. DOI: 10.1016/j.rser.2014.05.077. URL: http://www.sciencedirect.com/science/article/pii/S136403211400402X (visited on 07/27/2020).

[9] D. I. Tselentis, G. Yannis, and E. I. Vlahogianni. "Innovative motor insurance schemes: A review of current practices and emerging challenges." In: *Accident Analysis & Prevention* 98 (Jan. 2017), pp. 139–148. ISSN: 0001-4575. DOI: 10.1016/j.aap.2016.10.006. URL: http://www.sciencedirect.com/science/article/pii/S0001457516303670 (visited on 01/27/2018).

[10] T. A. Dingus, S. G. Klauer, V. L. Neale, A. Petersen, S. E. Lee, J. D. Sudweeks, M. A. Perez, J. Hankey, D. J. Ramsey, S. Gupta, C. Bucher, Z. R. Doerzaph, J. Jermeland, and R. R. Knipling. "The 100-Car Naturalistic Driving Study, Phase II - Results of the 100-Car Field Experiment." In: (Apr. 2006). URL: https://trid.trb.org/view/783477 (visited on 05/16/2018).

[11] O. Carsten, K. Kircher, and S. Jamson. "Vehicle-based studies of driving in the real world: The hard truth?" In: *Accident Analysis & Prevention* 58 (Sept. 2013), pp. 162–174. ISSN: 0001-4575. DOI: 10.1016/j.aap.2013.06.006. URL: http://www.sciencedirect.com/science/article/pii/S0001457513002340 (visited on 05/14/2018).

[12] D. A. Johnson and M. M. Trivedi. "Driving style recognition using a smartphone as a sensor platform." In: *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. Oct. 2011, pp. 1609–1615. DOI: 10.1109/ITSC.2011.6083078.

[13] J. Paefgen, F. Kehr, Y. Zhai, and F. Michahelles. "Driving Behavior Analysis with Smartphones: Insights from a Controlled Field Study." In: *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia*. MUM '12. New York, NY, USA: ACM, 2012, 36:1–36:8. ISBN: 978-1-4503-1815-0. DOI: 10.1145/2406367.2406412. URL: http://doi.acm.org/10.1145/2406367.2406412 (visited on 01/25/2018).

[14] S. Kantor and T. Stárek. "Design of Algorithms for Payment Telematics Systems Evaluating Driver's Driving Style." en. In: *Transactions on Transport Sciences* 7.1 (Mar. 2014), pp. 9–16. ISSN: 1802971X, 18029876. DOI: 10.2478/v10158-012-0049-5. URL: http://tots.upol.cz/doi/10.2478/v10158-012-0049-5.html (visited on 01/17/2018).

[15] H. Eren, S. Makinist, E. Akin, and A. Yilmaz. "Estimating driving behavior by a smartphone." In: *2012 IEEE Intelligent Vehicles Symposium*. June 2012, pp. 234–239. DOI: 10.1109/IVS.2012.6232298.

[16] K. Saleh, M. Hossny, and S. Nahavandi. "Driving behavior classification based on sensor data fusion using LSTM recurrent neural networks." In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. Oct. 2017, pp. 1–6. DOI: 10.1109/ITSC.2017.8317835.

[17] W. Weidner, F. W. G. Transchel, and R. Weidner. "Classification of scale-sensitive telematic observables for riskindividual pricing." en. In: *European Actuarial Journal* 6.1 (July 2016), pp. 3–24. ISSN: 2190-9733, 2190-9741. DOI: 10.1007/s13385-016-0127-x. URL: https://link.springer.com/article/10.1007/s13385-016-0127-x (visited on 12/06/2017).

[18] Y. L. Murphey, R. Milton, and L. Kiliaris. "Driver's style classification using jerk analysis." In: *2009 IEEE Workshop on Computational Intelligence in Vehicles and Vehicular Systems*. Mar. 2009, pp. 23–28. DOI: 10.1109/CIVVS.2009.4938719.

[19] J. Xie, A. R. Hilal, and D. Kulić. "Driving Maneuver Classification: A Comparison of Feature Extraction Methods." In: *IEEE Sensors Journal* 18.12 (June 2018), pp. 4777–4784. ISSN: 1530-437X. DOI: 10.1109/JSEN.2017.2780089.

[20] G. Singh, D. Bansal, and S. Sofat. "A smartphone based technique to monitor driving behavior using DTW and crowdsensing." In: *Pervasive and Mobile Computing* 40 (Sept. 2017), pp. 56–70. ISSN: 1574-1192. DOI: 10.1016/j.pmcj.2017.06.003. URL: http://www.sciencedirect.com/science/article/pii/S1574119216301250 (visited on 01/16/2019).

[21] C. Woo and D. Kulić. "Manoeuvre segmentation using smartphone sensors." In: *2016 IEEE Intelligent Vehicles Symposium (IV)*. June 2016, pp. 572–577. DOI: 10.1109/IVS.2016.7535444.

[22] Z. Camlica, A. Hilal, and D. Kulić. "Feature abstraction for driver behaviour detection with stacked sparse auto-encoders." In: *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. Oct. 2016, pp. 003299–003304. DOI: 10.1109/SMC.2016.7844743.

[23] M. Wu, S. Zhang, and Y. Dong. "A Novel Model-Based Driving Behavior Recognition System Using Motion Sensors." en. In: *Sensors* 16.10 (Oct. 2016), p. 1746. DOI: 10.3390/s16101746. URL: https://www.mdpi.com/1424-8220/16/10/1746 (visited on 01/16/2019).

[24] J. F. Júnior, E. Carvalho, B. V. Ferreira, C. d. Souza, Y. Suhara, A. Pentland, and G. Pessin. "Driver behavior profiling: An investigation with different smartphone sensors and machine learning." en. In: *PLOS ONE* 12.4 (2017), e0174959. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0174959. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0174959 (visited on 01/16/2019).

[25] E. Keogh and J. Lin. "Clustering of time-series subsequences is meaningless: implications for previous and future research." en. In: *Knowledge and Information Systems* 8.2 (Aug. 2005), pp. 154–177. ISSN: 0219-3116. DOI: 10.1007/s10115-004-0172-7. URL: https://doi.org/10.1007/s10115-004-0172-7 (visited on 12/08/2018).

[26] E. Romera, L. M. Bergasa, and R. Arroyo. "Need data for driver behaviour analysis? Presenting the public UAH-DriveSet." In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. Nov. 2016, pp. 387–392. DOI: 10.1109/ITSC.2016.7795584.

[27] L. M. Bergasa, D. Almería, J. Almazán, J. J. Yebes, and R. Arroyo. "DriveSafe: An app for alerting inattentive drivers and scoring driving behaviors." In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. June 2014, pp. 240–245. DOI: 10.1109/IVS.2014.6856461.

[28] E. Romera, L. M. Bergasa, and R. Arroyo. "A Real-Time Multi-scale Vehicle Detection and Tracking Approach for Smartphones." In: *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. Sept. 2015, pp. 1298–1303. DOI: 10.1109/ITSC.2015.213.

[29] F. Sagberg, S., G. Bianchi Piccinini, and J. Engström. "A Review of Research on Driving Styles and Road Safety." In: *Human factors* 57 (June 2015). DOI: 10.1177/0018720815591313.

[30] D. de Waard, C. Dijksterhuis, and K. A. Brookhuis. "Merging into heavy motorway traffic by young and elderly drivers." In: *Accident Analysis & Prevention* 41.3 (May 2009), pp. 588–597. ISSN: 0001-4575. DOI: 10.1016/j.aap.2009.02.011. URL: http://www.sciencedirect.com/science/article/pii/S0001457509000414 (visited on 05/13/2018).

[31] A. V. Desai and M. A. Haque. "Vigilance monitoring for operator safety: A simulation study on highway driving." In: *Journal of Safety Research* 37.2 (Jan. 2006), pp. 139–147. ISSN: 0022-4375. DOI: 10.1016/j.jsr.2005.11.003. URL: http://www.sciencedirect.com/science/article/pii/S0022437506000235 (visited on 05/13/2018).

[32] C. MacAdam, Z. Bareket, P. Fancher, and R. Ervin. "Using Neural Networks to Identify Driving Style and Headway Control Behavior of Drivers." In: *Vehicle System Dynamics* 29.sup1 (Jan. 1998), pp. 143–160. ISSN: 0042-3114. DOI: 10.1080/00423119808969557. URL: https://doi.org/10.1080/00423119808969557 (visited on 05/13/2018).

[33] T. A. Dingus, F. Guo, S. Lee, J. F. Antin, M. Perez, M. Buchanan-King, and J. Hankey. "Driver crash risk factors and prevalence evaluation using naturalistic driving data." en. In: *Proceedings of the National Academy of Sciences* 113.10 (Mar. 2016), pp. 2636–2641. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1513271113. URL: http://www.pnas.org/content/113/10/2636 (visited on 05/19/2018).

[34] M. Junger, R. West, and R. Timman. "Crime and Risky Behavior in Traffic: An Example of Cross-Situational Consistency." en. In: *Journal of Research in Crime and Delinquency* 38.4 (Nov. 2001), pp. 439–459. ISSN: 0022-4278. DOI: 10.1177/0022427801038004005. URL: https://doi.org/10.1177/0022427801038004005 (visited on 05/13/2018).

[35] D. Robertson, M. Winnett, and R. Herrod. "Acceleration signatures." In: *Traffic engineering and control* 33.9 (1992), pp. 485–491. ISSN: 0041-0683. URL: https://www.safetylit.org/citations/index.php?fuseaction=citations.viewdetails&citationIds[]=citjournalarticle_63339_19 (visited on 05/13/2018).

[36] A. E. A. Wåhlberg. "Driver celeration behaviour and accidents – an analysis." In: *Theoretical Issues in Ergonomics Science* 9.5 (Sept. 2008), pp. 383–403. ISSN: 1463-922X. DOI: 10.1080/14639220701596722. URL: https://doi.org/10.1080/14639220701596722 (visited on 05/13/2018).

[37] M. Dozza. "What factors influence drivers' response time for evasive maneuvers in real traffic?" In: *Accident Analysis & Prevention* 58 (Sept. 2013), pp. 299–308. ISSN: 0001-4575. DOI: 10.1016/j.aap.2012.06.003. URL: http://www.sciencedirect.com/science/article/pii/S0001457512002254 (visited on 05/19/2018).

[38] S. international. "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles." In: *SAE* (2018).

[39] M. R. Endsley and D. B. Kaber. "Level of automation effects on performance, situation awareness and workload in a dynamic control task." In: *Ergonomics* 42.3 (Mar. 1999). Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/001401399185595, pp. 462–492. ISSN: 0014-0139. DOI: 10.1080/001401399185595. URL: https://doi.org/10.1080/001401399185595 (visited on 10/12/2021).

[40] N. Merat and A. H. Jamson. "Is Drivers' Situation Awareness Influenced by a Fully Automated Driving Scenario?" In: *Human factors, security and safety*. Shaker Publishing. 2009.

[41] A. H. Jamson, N. Merat, O. M. Carsten, and F. C. Lai. "Behavioural changes in drivers experiencing highly-automated vehicle control in varying traffic conditions." In: *Transportation research part C: emerging technologies* 30 (2013), pp. 116–125.

[42] N. Merat, A. H. Jamson, F. C. Lai, M. Daly, and O. M. Carsten. "Transition to manual: Driver behaviour when resuming control from a highly automated vehicle." In: *Transportation research part F: traffic psychology and behaviour* 27 (2014), pp. 274–282.

[43] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh. "Experimental comparison of representation methods and distance measures for time series data." en. In: *Data Mining and Knowledge Discovery* 26.2 (Mar. 2013), pp. 275–309. ISSN: 1384-5810, 1573-756X. DOI: 10.1007/s10618-012-0250-5. URL: https://link.springer.com/article/10.1007/s10618-012-0250-5 (visited on 05/05/2018).

[44] J. Lin, E. Keogh, L. Wei, and S. Lonardi. "Experiencing SAX: a novel symbolic representation of time series." en. In: *Data Mining and Knowledge Discovery* 15.2 (Oct. 2007), pp. 107–144. ISSN: 1384-5810, 1573-756X. DOI: 10.1007/s10618-007-0064-z. URL: https://link.springer.com/article/10.1007/s10618-007-0064-z (visited on 05/15/2018).

[45] R. Agrawal, C. Faloutsos, and A. Swami. "Efficient similarity search in sequence databases." en. In: *Foundations of Data Organization and Algorithms*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Oct. 1993, pp. 69–84. ISBN: 978-3-540-57301-2 978-3-540-48047-1. DOI: 10.1007/3-540-57301-1_5. URL: https://link.springer.com/chapter/10.1007/3-540-57301-1_5 (visited on 05/15/2018).

[46] Y. Cai and R. Ng. "Indexing Spatio-temporal Trajectories with Chebyshev Polynomials." In: *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*. SIGMOD '04. New York, NY, USA: ACM, 2004, pp. 599–610. ISBN: 978-1-58113-859-7. DOI: 10.1145/1007568.1007636. URL: http://doi.acm.org/10.1145/1007568.1007636 (visited on 05/15/2018).

[47] K.-P. Chan and A. W.-C. Fu. "Efficient time series matching by wavelets." In: *Proceedings 15th International Conference on Data Engineering (Cat. No.99CB36337)*. Mar. 1999, pp. 126–133. DOI: 10.1109/ICDE.1999.754915.

[48] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. "Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases." en. In: *Knowledge and Information Systems* 3.3 (Aug. 2001), pp. 263–286. ISSN: 0219-1377. DOI: 10.1007/PL00011669. URL: https://link.springer.com/article/10.1007/PL00011669 (visited on 05/15/2018).

[49] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani. "Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases." In: *ACM Trans. Database Syst.* 27.2 (June 2002), pp. 188–228. ISSN: 0362-5915. DOI: 10.1145/568518.568520. URL: http://doi.acm.org/10.1145/568518.568520 (visited on 05/15/2018).

[50] F. Korn, H. V. Jagadish, and C. Faloutsos. "Efficiently Supporting Ad Hoc Queries in Large Datasets of Time Sequences." In: *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*. SIGMOD '97. New York, NY, USA: ACM, 1997, pp. 289–300. ISBN: 978-0-89791-911-1. DOI: 10.1145/253260.253332. URL: http://doi.acm.org/10.1145/253260.253332 (visited on 05/15/2018).

[51] Q. Chen, L. Chen, X. Lian, Y. Liu, and J. X. Yu. "Indexable PLA for Efficient Similarity Search." In: *Proceedings of the 33rd International Conference on Very Large Data Bases*. VLDB '07. Vienna, Austria: VLDB Endowment, 2007, pp. 435–446. ISBN: 978-1-59593-649-3. URL: http://dl.acm.org/citation.cfm?id=1325851.1325903 (visited on 05/15/2018).

[52] J. Serrà and J. L. Arcos. "An empirical evaluation of similarity measures for time series classification." In: *Knowledge-Based Systems* 67 (Sept. 2014), pp. 305–314. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2014.04.035. URL: http://www.sciencedirect.com/science/article/pii/S0950705114001658 (visited on 05/05/2018).

[53] G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. Smyth. "Rule Discovery from Time Series." In: *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*. KDD'98. New York, NY: AAAI Press, 1998, pp. 16–22. URL: http://dl.acm.org/citation.cfm?id=3000292.3000296 (visited on 05/16/2018).

[54] E. Frentzos, K. Gratsias, and Y. Theodoridis. "Index-based Most Similar Trajectory Search." In: *2007 IEEE 23rd International Conference on Data Engineering*. Apr. 2007, pp. 816–825. DOI: 10.1109/ICDE.2007.367927.

[55] D. J. Berndt and J. Clifford. "Using Dynamic Time Warping to Find Patterns in Time Series." In: *Proceedings of the AAAI Workshop on Knowledge Discovery in Databases*. 1994, pp. 359–370.

[56] M. Vlachos, G. Kollios, and D. Gunopulos. "Discovering similar multidimensional trajectories." In: *Proceedings 18th International Conference on Data Engineering*. 2002, pp. 673–684. DOI: 10.1109/ICDE.2002.994784.

[57] L. Chen and R. Ng. "On the Marriage of Lp-norms and Edit Distance." In: *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*. VLDB '04. Toronto, Canada: VLDB Endowment, 2004, pp. 792–803. ISBN: 978-0-12-088469-8. URL: http://dl.acm.org/citation.cfm?id=1316689.1316758 (visited on 05/16/2018).

[58] L. Chen, M. T. Özsu, and V. Oria. "Robust and Fast Similarity Search for Moving Object Trajectories." In: *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*. SIGMOD '05. New York, NY, USA: ACM, 2005, pp. 491–502. ISBN: 978-1-59593-060-6. DOI: 10.1145/1066157.1066213. URL: http://doi.acm.org/10.1145/1066157.1066213 (visited on 05/16/2018).

[59] M. D. Morse and J. M. Patel. "An Efficient and Accurate Method for Evaluating Time Series Similarity." In: *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*. SIGMOD '07. New York, NY, USA: ACM, 2007, pp. 569–580. ISBN: 978-1-59593-686-8. DOI: 10.1145/1247480.1247544. URL: http://doi.acm.org/10.1145/1247480.1247544 (visited on 05/16/2018).

[60] P. F. Marteau. "Time Warp Edit Distance with Stiffness Adjustment for Time Series Matching." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.2 (Feb. 2009), pp. 306–318. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2008.76.

[61] J. Serrà and J. L. Arcos. "A Competitive Measure to Assess the Similarity between Two Time Series." en. In: *Case-Based Reasoning Research and Development*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Sept. 2012, pp. 414–427. ISBN: 978-3-642-32985-2 978-3-642-32986-9. DOI: 10.1007/978-3-642-32986-9_31. URL: https://link.springer.com/chapter/10.1007/978-3-642-32986-9_31 (visited on 05/16/2018).

[62] Y. Chen, M. A. Nascimento, B. C. Ooi, and A. K. H. Tung. "SpADe: On Shape-based Pattern Detection in Streaming Time Series." In: *2007 IEEE 23rd International Conference on Data Engineering*. Apr. 2007, pp. 786–795. DOI: 10.1109/ICDE.2007.367924.

[63] J. Aßfalg, H.-P. Kriegel, P. Kröger, P. Kunath, A. Pryakhin, and M. Renz. "Similarity Search on Time Series Based on Threshold Queries." en. In: *Advances in Database Technology - EDBT 2006*. Lecture Notes in Computer Science.

Springer, Berlin, Heidelberg, Mar. 2006, pp. 276–294. ISBN: 978-3-540-32960-2 978-3-540-32961-9. DOI: 10.1007/11687238_19. URL: https://link.springer.com/chapter/10.1007/11687238_19 (visited on 05/16/2018).

[64] V. I. Levenshtein. "Binary Codes Capable of Correcting Deletions, Insertions and Reversals." In: *Soviet Physics Doklady* 10 (Feb. 1966), p. 707. URL: http://adsabs.harvard.edu/abs/1966SPhD...10..707L (visited on 05/16/2018).

[65] J. Lin, E. Keogh, S. Lonardi, and P. Patel. "Finding Motifs in Time Series." In: *Proceedings of the Second Workshop on Temporal Data Mining*. July 2002, pp. 53–68. URL: http://citeseer.ist.psu.edu/lin02finding.html (visited on 02/03/2019).

[66] Y. Tanaka, K. Iwamoto, and K. Uehara. "Discovery of Time-Series Motif from Multi-Dimensional Data Based on MDL Principle." en. In: *Machine Learning* 58.2 (Feb. 2005), pp. 269–300. ISSN: 1573-0565. DOI: 10.1007/s10994-005-5829-2. URL: https://doi.org/10.1007/s10994-005-5829-2 (visited on 01/22/2019).

[67] S. Torkamani and V. Lohweg. "Survey on time series motif discovery." en. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 7.2 (2017), e1199. ISSN: 1942-4795. DOI: 10.1002/widm.1199. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1199 (visited on 01/22/2019).

[68] A. Mueen. "Time series motif discovery: dimensions and applications." en. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 4.2 (2014), pp. 152–159. ISSN: 1942-4795. DOI: 10.1002/widm.1119. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1119 (visited on 01/22/2019).

[69] 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: *2016 IEEE 16th International Conference on Data Mining (ICDM)*. ISSN: 2374-8486. Dec. 2016, pp. 1317–1322. DOI: 10.1109/ICDM.2016.0179.

[70] D. Minnen, C. Isbell, I. Essa, and T. Starner. "Detecting Subdimensional Motifs: An Efficient Algorithm for Generalized Multivariate Pattern Discovery." In: *Seventh IEEE International Conference on Data Mining (ICDM 2007)*. Oct. 2007, pp. 601–606. DOI: 10.1109/ICDM.2007.52.

[71] 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. URL: http://dl.acm.org/citation.cfm?id=1661445.1661647 (visited on 01/25/2019).

[72] A. Balasubramanian, J. Wang, and B. Prabhakaran. "Discovering Multidimensional Motifs in Physiological Signals for Personalized Healthcare." In: *IEEE Journal of Selected Topics in Signal Processing* 10.5 (Aug. 2016), pp. 832–841. ISSN: 1932-4553. DOI: 10.1109/JSTSP.2016.2543679.

[73] B. Liu, Y. Liu, J. Li, J. Lang, and R. Gu. "Multi-dimensional motif discovery in air pollution data." In: *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. Oct. 2017, pp. 531–536. DOI: 10.1109/SMC.2017.8122660.

[74] P. Nunthanid, V. Niennattrakul, and C. A. Ratanamahatana. "Parameter-free motif discovery for time series data." In: *2012 9th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*. May 2012, pp. 1–4. DOI: 10.1109/ECTICon.2012.6254126.

[75] Y. Gao and J. Lin. "Exploring variable-length time series motifs in one hundred million length scale." en. In: *Data Mining and Knowledge Discovery* 32.5 (Sept. 2018), pp. 1200–1228. ISSN: 1573-756X. DOI: 10.1007/s10618-018-0570-1. URL: https://doi.org/10.1007/s10618-018-0570-1 (visited on 01/25/2019).

[76] J. Rissanen. *Stochastic Complexity In Statistical Inquiry*. en. Google-Books-ID: KY4GCwAAQBAJ. World Scientific, Oct. 1998. ISBN: 978-981-4507-40-0.

[77] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise." en. In: *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*. 1996, pp. 226–231.

[78] M. C. Hao, M. Marwah, H. Janetzko, U. Dayal, D. A. Keim, D. Patnaik, N. Ramakrishnan, and R. K. Sharma. "Visual exploration of frequent patterns in multivariate time series." en. In: *Information Visualization* 11.1 (Jan. 2012), pp. 71–83. ISSN: 1473-8716. DOI: 10.1177/1473871611430769. URL: https://journals.sagepub.com/doi/abs/10.1177/1473871611430769 (visited on 01/28/2020).

[79] A. Balasubramanian and B. Prabhakaran. "Flexible exploration and visualization of motifs in biomedical sensor data." In: *Proc. of Workshop on Data Mining for Healthcare, in conjunction with ACM KDD*. 2013.

[80] P. Senin, J. Lin, X. Wang, T. Oates, S. Gandhi, A. P. Boedihardjo, C. Chen, S. Frankenstein, and M. Lerner. "GrammarViz 2.0: A Tool for Grammar-Based Pattern Discovery in Time Series." en. In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by T. Calders, F. Esposito, E. Hüllermeier, and R. Meo. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2014,

pp. 468–472. ISBN: 978-3-662-44845-8. DOI: 10.1007/978-3-662-44845-8_37.

[81] T Kohonen. *Self-Organizing Maps*. 3rd editio. Berlin: Springer, 2001.

[82] A. Mueen, E. Keogh, Q. Zhu, S. Cash, and B. Westover. "Exact Discovery of Time Series Motifs." In: *Proceedings of the 2009 SIAM International Conference on Data Mining*. Proceedings. Society for Industrial and Applied Mathematics, Apr. 2009, pp. 473–484. ISBN: 978-0-89871-682-5. DOI: 10.1137/1.9781611972795.41. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611972795.41 (visited on 07/28/2020).

[83] J. Lin and Y. Li. "Finding approximate frequent patterns in streaming medical data." In: *2010 IEEE 23rd International Symposium on Computer-Based Medical Systems (CBMS)*. ISSN: 1063-7125. Oct. 2010, pp. 13–18. DOI: 10.1109/CBMS.2010.6042675.

[84] R Henriques, F Bacao, and V Lobo. "Artificial Intelligence in Geospatial Analysis: applications of Self-Organizing Maps in the context of Geographic Information Science." Doctoral dissertation. Lisboa, 2010. URL: http://hdl.handle.net/10362/5723.

[85] S Kaski and T Kohonen. "Exploratory data analysis by the self-organizing map: structures of welfare and poverty in the world." In: *Neural Networks in Financial Engineering*. Ed. by N Apostolos-Paul, Yaser Refenes, Yaser Abu-Mostafa, John Moody, and A. Weigend. Singapore: World Scientific, 1996, pp. 498–507. URL: http://www.cis.hut.fi/{~}sami/therest.html.

[86] S Kaski, J Nikkilä, and T Kohonen. "Methods for interpreting a self-organized map in data analysis." In: *Proceedings of ESANN'98 , 6th European Symposium on Artificial Neural Networks*. Ed. by M. Verleysen. Bruges, Belgium: D-Facto, 1998, pp. 185–190. URL: http://www.cis.hut.fi/{~}sami/therest.html.

[87] R. Henriques, F. BaÇão, and V. Lobo. "Carto-SOM: cartogram creation using self-organizing maps." In: *International Journal of Geographical Information Science* 23.4 (2009), pp. 483–511. ISSN: 1365-8816. DOI: 10.1080/13658810801958885. URL: https://www.tandfonline.com/doi/abs/10.1080/13658810801958885.

[88] J Vesanto. "SOM-based data visualization methods." In: *Intelligent Data Analysis* 3 (1999), pp. 111–126. URL: https://dl.acm.org/doi/10.1016/S1088-467X\%2899\%2900013-X.

[89] A Ultsch and H. P. Siemon. "Kohonen Networks on Transputers: Implementation and Animation." In: *International Neural Network Conference (INNC)*. 1990.

[90] D. Wang and S. Tapan. "A Robust Elicitation Algorithm for Discovering DNA Motifs Using Fuzzy Self-Organizing Maps." In: *IEEE Transactions on Neural Networks and Learning Systems* 24.10 (2013), pp. 1677–1688. ISSN: 2162-2388. DOI: 10.1109/TNNLS.2013.2275733.

[91] H. F. Bassani and A. F. R. Araujo. "Dimension Selective Self-Organizing Maps With Time-Varying Structure for Subspace and Projected Clustering." In: *IEEE Transactions on Neural Networks and Learning Systems* 26.3 (2015), pp. 458–471.

[92] R. C. Brito and H. F. Bassani. "Self-Organizing Maps with Variable Input Length for Motif Discovery and Word Segmentation." In: *2018 International Joint Conference on Neural Networks (IJCNN)* (2018). DOI: 10.1109/ijcnn.2018.8489090. URL: http://dx.doi.org/10.1109/IJCNN.2018.8489090.

[93] A. Novikov. "PyClustering: Data Mining Library." In: *Journal of Open Source Software* 4.36 (2019), p. 1230. DOI: 10.21105/joss.01230. URL: https://doi.org/10.21105/joss.01230.

[94] H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, and Hexagon-ML. *The UCR Time Series Classification Archive*. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/. 2018.

[95] 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*. Society for Industrial and Applied Mathematics, Apr. 2004. DOI: 10.1137/1.9781611972740.2. URL: https://doi.org/10.1137/1.9781611972740.2.

[96] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan. "uWave: Accelerometer-based personalized gesture recognition and its applications." en. In: *Pervasive and Mobile Computing*. PerCom 2009 5.6 (Dec. 2009), pp. 657–675. ISSN: 1574-1192. DOI: 10.1016/j.pmcj.2009.07.007. URL: http://www.sciencedirect.com/science/article/pii/S1574119209000674 (visited on 04/09/2020).

[97] E. Velenis, P. Tsiotras, and J. Lu. "Optimality Properties and Driver Input Parameterization for Trail-braking Cornering." en. In: *European Journal of Control* 14.4 (Jan. 2008), pp. 308–320. ISSN: 0947-3580. DOI: 10.3166/ejc.14.308-320. URL: http://www.sciencedirect.com/science/article/pii/S0947358008707751 (visited on 12/01/2020).

[98] I. Škrjanc, G. Andonovski, A. Ledezma, O. Sipele, J. A. Iglesias, and A. Sanchis. "Evolving cloud-based system for the recognition of drivers' actions." en. In: *Expert Systems with Applications* 99 (June 2018), pp. 231–238. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2017.11.008. URL: http://www.sciencedirect.com/science/article/pii/S0957417417307583 (visited on 12/01/2020).

[99] M. I. Silva and R. Henriques. "Finding manoeuvre motifs in vehicle telematics." en. In: *Accident Analysis & Prevention* 138 (Apr. 2020), p. 105467. ISSN: 0001-4575. DOI: 10.1016/j.aap.2020.105467. URL: http://www.sciencedirect.com/science/article/pii/S0001457519304154 (visited on 02/23/2020).

[100] S. Jain, D. Hallac, R. Sosic, and J. Leskovec. "MASA: Motif-Aware State Assignment in Noisy Time Series Data." In: *arXiv:1809.01819 [cs, stat]*. arXiv: 1809.01819. Anchorage, Alaska, USA, Aug. 2019. URL: http://arxiv.org/abs/1809.01819 (visited on 03/14/2020).

[101] C. Schwarz. "Time Series Categorization of Driving Maneuvers Using Acceleration Signals." en. In: *Proceedings of the 9th International Driving Symposium on Human Factors in Driver Assessment, Training, and Vehicle Design: driving assessment 2017*. Manchester Village, Vermont, USA: University of Iowa, 2017, pp. 270–276. DOI: 10.17077/drivingassessment.1646. URL: http://ir.uiowa.edu/drivingassessment/2017/papers/41 (visited on 03/14/2020).

[102] C.-C. M. Yeh, N. Kavantzas, and E. Keogh. "Matrix Profile VI: Meaningful Multidimensional Motif Discovery." In: *2017 IEEE International Conference on Data Mining (ICDM)*. ISSN: 2374-8486. Nov. 2017, pp. 565–574. DOI: 10.1109/ICDM.2017.66.

[103] M. I. Silva and R. Henriques. "Exploring time-series motifs through DTW-SOM." In: *arXiv:2004.08176 [cs, stat]*. arXiv: 2004.08176. Apr. 2020. URL: http://arxiv.org/abs/2004.08176 (visited on 06/27/2020).

[104] T. Kohonen. *Self-Organizing Maps*. en. 3rd ed. Springer Series in Information Sciences. Berlin Heidelberg: Springer-Verlag, 2001. ISBN: 978-3-540-67921-9. DOI: 10.1007/978-3-642-56927-2. URL: https://www.springer.com/gp/book/9783540679219 (visited on 06/27/2020).