Mining Sensor Data from Complex Systems

Proefschrift

ter verkrijging van
de graad van Doctor aan de Universiteit Leiden,
op gezag van Rector Magnificus prof.mr. C.J.J.M. Stolker,
volgens besluit van het College voor Promoties
te verdedigen op dinsdag 15 December 2015
klokke 12.30 uur

door

Ugo Vespier

geboren te Lamezia Terme
in 1985

Promotiecommissie

Promotor:	prof. dr. J. N. Kok	
Co-promotor:	dr. A. J. Knobbe	
Overige leden:	prof. dr. E. Keogh	(University of California, Riverside)
	dr. J. Gama	(University of Porto)
	prof. dr. C. Rieffe	
	prof. dr. A. Plaat	

Alla mia famiglia.

Contents

Chapter 1

Introduction

Over the last decades, the advances in computational power, storage technology and sensor networks have made data an abundant resource [79]. Today, virtually everything, from natural phenomena to complex artificial and physical systems, can be measured and the resulting information collected, stored and analyzed in order to gain new insight, optimize existing processes or both.

The term *Big Data* has gained popularity, in academia, industry and the public opinion, to describe the opportunities and the challenges connected to this huge explosion in data availability [64]. In a report by IDC [33], the authors estimate that the total amount of data in the digital world will amount to 40.000 exabytes[1] by the end of 2020, almost doubling its size every year. The data sources are diverse, ranging from user-contributed material on social networks (i.e. posts, tweets and status updates) to consumer behavioral data collected by online retailers such as product views and purchases.

In particular, advances in measuring technology and sensors networks [3, 16] greatly contributed to the explosion of data. The adoption and deployment of measurement systems for all sorts of industrial, commercial and consumer applications, is paving the way to important opportunities for monitoring and analyzing all kinds of systems over time at a level of detail never experienced

[1] This equates to $4 \cdot 10^{22}$ bytes.

before.

In fact, sensing technology and the ability to manage big data represent a fundamental improvement in our ability to measure complex systems. Multiple types of sensors, high sampling rates, advances in noise reduction techniques, to cite a few, are all improvements that are contributing to making progressively better representations of systems in data.

As a result, new challenges in the analysis and visualization of this large amount of sensor data have emerged and, over the last decade, the efforts of the research community to provide solutions to these problems have soared [1]. Methods and algorithms, in fact, will have to advance in order to cope with the increased complexity of the time series datasets available and to improve the ability to learn from the greater level of detail present in them.

A side effect of the exponential explosion of data collection is that *labeled* information will be an increasingly scarce resource in the future, as it is extremely costly to produce labeled datasets in relation to the current rate of data growth. Because of this, the task of extracting structured information from unlabeled data is of paramount importance when dealing with the challenges posed by big data. The algorithms and methods presented in this thesis are designed to work in this scenario where novel insights have to be extracted in an unsupervised way.

In particular, the focus of this thesis is the analysis of complex sensor data in the form of *time series*. Time series are sequences of observations sampled periodically over time. We approach the analysis of such data from a *data mining* perspective, with the end goal of extracting previously unknown knowledge and insight in the data. Data mining [37, 104] (DM) is a discipline aimed at discovering useful and structured patterns in large collections of data. Data mining methods lie at the intersection between computer science, machine learning, database systems and statistics, and are a fundamental part of any KDD (Knowledge Discovery in Databases) process [27]. Time series, on the other hand, are a ubiquitous type of data, and mining time series data represents an important branch of DM.

In this thesis, we introduce data mining and visualization methods for large time series data collected from complex physical systems by means of sensors.

Our work is motivated by InfraWatch [55], a Structural Health Monitoring project centered around the management and analysis of data collected by a large sensor network deployed on a highway bridge. The sensor network comprises strain, temperature and vibration sensors, sampling continuously at 100 Hz. A highway bridge is a complex system and so is the data collected. The behaviour of the bridge, and consequently the properties of the data, is affected by external factors such as the temperature, weather conditions, traffic activity and deterioration of the concrete. Moreover, InfraWatch data contains repeated patterns at different resolutions due to the bridge's response to recurring events such as passing vehicles or traffic jams. Because of these characteristics, InfraWatch is an ideal testbed for evaluating the methods we introduce.

In this thesis, we will develop and discuss solutions to the following fundamental questions when dealing with large and complex time-series data collected from sensors like the one provided by InfraWatch:

- What are the relevant temporal scales of analysis for a given time series?
- Which are the recurring multi-scale patterns present in a given time series?
- How can we effectively model and recognize events in time series data?
- How can we support efficient and interactive visualization of massive time series data?

Complex systems are often affected by several phenomena at multiple temporal scales and this effect is reflected in the collected data. Consider, for example, the time series produced by one of the strain sensors of the InfraWatch bridge. This data is the result of the superimposed effects of the passing vehicles, traffic jams and more long-term effects such as the day-night cycle in temperature, which in turn affect the response of the structure. Throughout the thesis, we will introduce a method to discover which are the relevant

temporal scales of analysis and introduce a decomposition of the original time series such that every component represents a single phenomenon at its characteristic scale. The goal of the method is to find the underlying factors that explain the input data.

These multiple phenomena, moreover, are often characterized by the presence of patterns that repeat over time and reflect their effect in the data. Consider again the strain sensor example. The effect of traffic jams will produce similar recurring patterns in the data, for example every morning during rush hour. The same would happen with the effect of passing vehicles, although they will appear at a shorter time scale (in the order of seconds) and potentially superimposed on the traffic jam patterns. We will introduce a method to mine recurring patterns, so-called *motifs* in the literature, from time series data at multiple temporal scales.

The third research question addresses the problem of clustering time series subsequences in order to model and recognize fixed-length events in the data. Time series clustering has proven to be a difficult task, as it is hard to model the subsequence space properly without introducing artifacts in the results [49]. We will introduce a novel distance measure to cope with this problem.

Finally, we will address the problem of massive time series visualization. Although visualization is not directly related to data mining, it is a fundamental task in every data science project, especially to support the exploratory phases and build an idea of the data at hand. When exploring and visualizing a dataset, interactivity is important as it permits testing ideas and assumptions quickly without having to wait excessive periods. We will see how we made the interactive visualization of terabyte-sized datasets possible by introducing an ad-hoc storage scheme for time series, which effectiveness has been proven by a real world software package called VizTool.

Although these research questions find a natural application in the InfraWatch project for the analysis of bridge sensor data, we stress that they are instrumental to the understanding of many complex systems. In fact, the presence in the data of multiple temporal scales and recurring phenomena, as well as

the need for effective visualization, are general challenges shared among all complex physical and artificial systems measured by sensor networks.

The methods and algorithms introduced in this thesis combine concepts from data mining, signal processing, and information theory. In particular, in order to formally characterize the concept of temporal scales, we will make use of concepts from the field of signal processing, such as the theory of scale-space [103, 62].

As we are interested in extracting new insights from the data, such as the relevant temporal scales and the recurring events, we approach the problem from a *compression* standpoint. The idea of using concepts from the theory of compression in order to learn new facts about the structure of a dataset has been widely considered and explored in the literature [82, 105]. Data compression techniques geared towards learning have been employed for categorizing text [29], clustering data [17], devising similarity measures [99, 52], in genomic analysis [36], data discretization [57], pattern mining [100, 68, 84], stream mining [89], and as the base of parameter-free data mining methods [51].

We will see how we can define parametrized compression schemes for time series in order to find the one that best compresses the data and, at the same time, results as simple as possible. We employ the Minimum Description Length framework [34], a formalization of the Occam's Razor principle [91], in order to select the best compression model among the many possible and conceptually discern what is notable, and what can be ignored, in the data. We will see how this approach deals with many of the challenges present when analyzing real-world time series data, such as the presence of noisy measurements, the occurrence of spurious and anomalous events and, ultimately, the risk of over-fitting the data with models that would be hardly general.

1.1 Thesis Outline

Below, we give a brief outline of the dissertation, summarizing the contents of the following chapters. As most chapters are based on previous publications by the author, we also give the appropriate references to them when this is the case.

In **Chapter 2: Sensor Data and Applications**, we give the main motivations behind this work and introduce the InfraWatch project.

In **Chapter 3: Preliminaries and Background**, we introduce fundamental material and concepts that will be used throughout the rest of the thesis, especially in Chapter 4 and Chapter 5.

In **Chapter 4: Identifying the Relevant Temporal Scales**, we discuss a method for discovering the most relevant scales of analysis, and their corresponding scale components, in time series data. This work was published in the following paper [96]:

> Vespier U., Knobbe A., Nijssen S., and Vanschoren J., *MDL-based Analysis of Time Series at Multiple Time-Scales*, in Proceedings ECML-PKDD 2012, Bristol, UK.

This work is also part of the following book chapter [95]:

> Vanschoren, J., Vespier, U., Miao, S., Meeng, M., Cachucho, R., Knobbe, A., *Large-scale sensor network analysis — Applications in structural health monitoring*, in Big Data Management, Technologies, and Applications, IGI Global, 2013

In **Chapter 5: Mining Variable-Length Motifs at Multiple Scales**, we introduce a method for mining variable-length, and potentially overlapping, motifs at multiple temporal scales in sensor-based time series data. This work was published in the following paper [98]:

> Vespier, U., Knobbe, A., Nijssen, S., *Mining Characteristic Multi-Scale Motifs in Sensor-Based Time Series*, in Proceedings CIKM 2013, San Francisco, USA.

In **Chapter 6: Subsequences Clustering for Events Modeling**, we discuss a distance measure and an associated method for the effective clustering time-series subsequences for events modeling. This work was published in the following paper [97]:

> Vespier, U., Knobbe, A., Vanschoren, J., Miao, S., Koopman, A., Obladen, B., Bosma, C., *Traffic Events Modeling for Structural Health Monitoring*, in Proceedings IDA 2011, Porto, Portugal.

This work is also part of a book chapter [95]:

> Vanschoren, J., Vespier, U., Miao, S., Meeng, M., Cachucho, R., Knobbe, A., *Large-scale sensor network analysis — Applications in structural health monitoring*, in Big Data Management, Technologies, and Applications, IGI Global, 2013

In **Chapter 7: Interactive Time-Series Visualization**, we introduce a method and a software platform for visualizing terabyte sized time-series dataset. This work was published in the following paper [6]:

> Baggio, A., Vespier, U., Knobbe, A., *Automated Selection of Data-Adaptive Approximations for Large Time-Series Visualization*, in Proceedings Benelearn 2013, Nijmegen, the Netherlands

In **Chapter 8: Conclusions**, we draw the overall conclusions regarding this work and highlight some final considerations about its impact and potential future work.

Chapter 2

Sensor Data and Complex Systems

In this chapter, we discuss the context of this dissertation and the motivation behind the presented work. We also present the InfraWatch project, the main application and testbed for the methods discussed in the next chapters.

2.1 Big Data

As mentioned, the term *Big Data* has received a lot of attention over the last years [64], although it is often cause of confusion as its meaning is not always well-defined in all contexts. Big data is a broad term that refers to the challenges in managing and analyzing large quantities of data.

A widely accepted definition of Big Data has been given by Gartner's analysts Beyer and Laney in a 2012 industry report [11] where the authors define the term in relation to three main challenges:

Volume The ever-growing amount of data that institutions and companies have to deal with offers serious challenges in terms of data management and storage. Real-world examples range from the data collected by astronomers with radio telescopes, to the massive amount of messages

exchanged nowadays on social networking platforms such as Facebook and Twitter. Novel storage and indexing methods, able to cope with this huge amount of collected information, need to be employed.

Velocity A second important challenge is related to how long it takes to process incoming data. For example, a credit card institution analyzing massive streams of incoming transactions would like to detect potential fraud without delay. Real-time processing methods are needed in order to cope with the *velocity* challenge.

Variety Last but not least, Big Data comes in any type, both in structured and unstructured form. Text, audio, video, sensor data, log files, and combinations of these, are examples of data types found in Big Data applications [15]. The development of methods able to cope with this broad variety of data is another challenge posed by Big Data applications.

A great extent of the efforts aimed at solving big data problems revolve around these three challenges, as well as a broad range of applications across several science fields and industries. Sensor networks and monitoring is an important one and represents the main focus of this thesis. In particular, the rise of the Internet of Things [4, 5] is directly connected to the challenges posed by the management and analysis of Big Data.

2.2 Sensor Networks and the Internet of Things

The last two decades have witnessed a tremendous growth in the availability of sensor data collected from a multitude of systems in various application domains [3, 16]. In fact, the wider availability of cheap sensor technology has enabled large sensor networks that continuously monitor and analyze physical systems such as infrastructures, cars [25, 45, 59, 32], airplanes [7, 107, 9, 87] and, last but not least, the human body [14, 83].

The increasing presence of sensing devices coupled with the pervasiveness of connectivity is paving the way for a scenario in which physical objects,

humans and software communicate to achieve common goals by directly interacting with the physical world. This paradigm is called Internet of Things (IoT) [35, 5] and several applications of its concepts are already used in production.

In a recent report [72] by McKinsey & Company, the authors categorize IoT applications in three main areas:

- Tracking behavior
- Enhanced situational awareness
- Sensor-driven decision analytics

Companies and institutions are interested, first of all, in tracking and monitoring products and objects in real-time [54] in order to increase the efficiency of their operations or fine-tune their business or pricing models. Consider, for example, the case of a car insurance company that installs sensors in their customers' cars in order to monitor and model the behavior of drivers [90].

Situational awareness [106] is another key application area of the IoT. Large sensors networks, in fact, can be deployed in infrastructures, such as roads, buildings or bridges, or installed in certain areas to report on environmental conditions. Data coming from the sensors can then be used to enhance the awareness of decision makers about the observed events in real-time, especially when data is coupled with tailored visualization technologies.

Ultimately, sensor networks can support long-term and complex decision making. In the retail industry, for example, some companies are experimenting with sensors that continuously monitor shoppers [67, 63] as they move through stores in order to measure how long they stand in front of any given display and correlate it with what they ultimately buy. Data of this kind can help, in the long run, by optimizing store layouts and increase revenues.

Especially when long-term decision support is of interest, sensors networks produce large volumes of data continuously over time. This opens up several challenges both from a data management and from a data analysis perspective. Modern sensor networks have to be supported with state-of-the-art data

management solutions in order to cope with the large amount of collected data and ensure its effective storage, access and visualization.

Such large amounts of data, on the other hand, represent an opportunity to apply data mining methods to better understand the observed system and get insight into its behavior [1]. Moreover, as these data sources continuously provide data over time, the research community is also focusing on methods and algorithms to analyze information in a streaming fashion in order to provide real-time insights [31].

2.3 Multi-Scale nature of Complex Systems

Sensor networks are often employed to monitor and analyze complex, dynamic systems, which exhibit non-obvious behavior. An important example are systems affected by several phenomena at different temporal scales.

Consider, for example, the electrical system of an apartment whose aggregate power consumption is measured by a smart meter [108, 58]. The time series data collected by the smart meter would be affected by all the operational home appliances, heating units and lighting systems in the house. As different home appliances are switched on at different times and have diverse operational durations, their effect on the aggregate consumption can range from short-term spikes, for example in the case of a boiler, to longer, more equally distributed patterns, as for example in the case of a washing machine [73]. For example, the time series in Figure 2.1 shows four days of power usage from an apartment in the Smart* dataset [8]. In the data, there is a clear long-term periodic component due to the cycles of the refrigerator. Shorter-term patterns, however, show up superimposed in the data and correspond to activations and deactivations of the various electrical appliances in the house.

Another real-world example borrowed from civil engineering regards dikes [20, 70]. In dikes, it is typically of interest for civil engineers to measure the water pressure at specified location of the infrastructure. The amount of pressure depends on several factors and, also in this case, it is possible to classify them

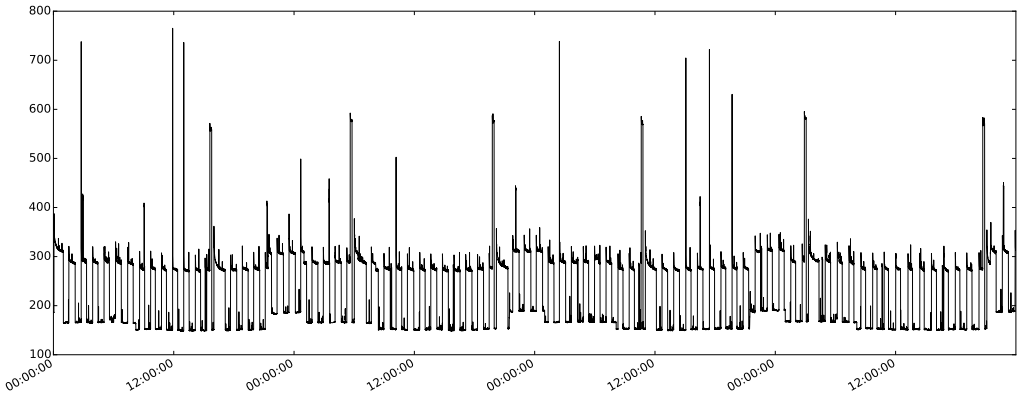


Figure 2.1: Four full days of power usage (in watts) from one of the houses in the Smart* dataset [8].

along diverse temporal scales. For example, lunar tides indirectly affects water pressure following both a half-daily cycle and a longer-term, two-weekly cycle.

Analysing systems such as the ones described above requires methods capable of dealing with the presence of multiple relevant scales of analysis in order to extract insights at all levels and resolutions. This represents one of the main challenges addressed in this thesis.

2.4 SHM and InfraWatch

One relatively recent application of sensor networks and sensor data analysis is the monitoring of infrastructural assets such as bridges, tunnels, etc. [21]. In fact, according to a recent survey from the US Federal Highway Commission [28], on average 56% of the assessments to civil infrastructures made by visual inspection are inappropriate, suggesting additional methods of monitoring to guarantee the safety of the assets.

Structural Health Monitoring (SHM) is an interdisciplinary field at the intersection between civil engineering, signal processing, sensor technology, material sciences, data management and mining, which is emerging in order to find alternative or complementary solutions to the visual inspection. In

fact, the use of advanced sensing and monitoring systems provides the opportunity to collect real-time information from infrastructures, in order to monitor their performance and to deduce relevant knowledge for decisions on their maintenance demand [26, 86]. Asset owners can use this information to assess the life time perspective of (crucial) infrastructural links and to plan the window within which maintenance can be conducted. When considering the stock of infrastructural assets in view of service-life assessment, monitoring and sensing systems are very valuable instruments that can be used to extract actual information about its condition and performance.

In typical SHM scenarios, sensor systems are mounted in or to structures and monitor the environmental as well as the internal condition of the measured system over long periods of time. The collected sensor data is typically continuously analyzed in order to detect inconsistencies or anomalies in how the structure is behaving and notify potential problems in time. Aside from notifying anomalies, SHM systems are also used to monitor and forecast degradation mechanisms in order to plan maintenance in a more informed way.

In the next section, we present a particular SHM project in detail. This project and its data will serve as a testbed for a great extent of the methods and algorithms presented in this thesis.

2.4.1 The InfraWatch project

InfraWatch is a project that is part of a Dutch STW's funded program called Integral Solutions for Sustainable Construction (IS2C). The program is composed of nine research projects with the common goal of setting new standards and advancing the state of the art in the field of sustainable construction and service-life assessment.

As part of the IS2C program, InfraWatch[1] focuses on sensing, monitoring and degradation mechanism from a data analysis perspective. Subject of the project is an important Dutch highway bridge: the Hollandse Brug. The Hol-

[1] http://www.infrawatch.com

Figure 2.2: Picture of the Hollandse Brug, which connects the 'island' Flevoland to the province Noord-Holland.

landse Brug is a bridge between the Flevoland and Noord-Holland provinces and is located at the place where the Gooimeer joins the IJmeer (see Figure 2.2). The bridge was opened in June 1969 and National Road A6 uses it. There is also a rail connection parallel to the highway bridge, as well as a lane for cyclists on the west side of the car bridge.

In April 2007, it was announced that measurements would have shown that the bridge did not meet the quality and security requirements. Therefore, the bridge was closed in both directions for heavy traffic on April 27, 2007. The repairs were launched in August 2007 and a consortium of companies, Strukton, RWS and Reef has installed a monitoring configuration underneath the first south span of the Hollandse Brug with the main aim to collect data for evaluating how the bridge responds to load. The sensor network is part of the strengthening project which was necessary to upgrade the bridge's capacity by overlaying.

The monitoring system comprises 145 sensors that measure different aspects of the condition of the bridge, at several locations along the bridge (see Figure 2.3 for an illustration). The following types of sensors are employed [55, 56]:

- 34 'geo-phones' (vibration sensors) that measure the vertical movement of the bottom of the road-deck as well as the supporting columns.
- 16 strain gauges embedded in the concrete, measuring horizontal longitudinal strain, and an additional 34 gauges attached to the outside.
- 28 strain gauges embedded in the concrete, measuring horizontal strain perpendicular to the first 16 strain gauges, and an additional 13 gauges attached to the outside.
- 10 thermometers embedded in the concrete, and 10 attached on the outside.

Furthermore, there is a weather station, and a video-camera provides a continuous video stream of the actual traffic on the bridge. Additionally, there are also plans to monitor the adjacent railway bridge.

The current monitoring set-up is clearly providing many challenges for data management. The 145 sensors are in fact producing data at rates of 100 Hz, which can amount to a gigabyte of data per day. Adding to that is the continuous stream of video.

Project goals and expectations

InfraWatch is, primarily, a Structural Health Monitoring project and its goals are directly related to questions about the observed infrastructure from a civil engineering perspective. The following tasks, in particular, are of importance to the civil engineers involved in the project:

- obtain a summary of the major phenomena affecting the bridge infrastructure over time and their impact.
- given historical sensor data from the bridge, obtain a qualitative and

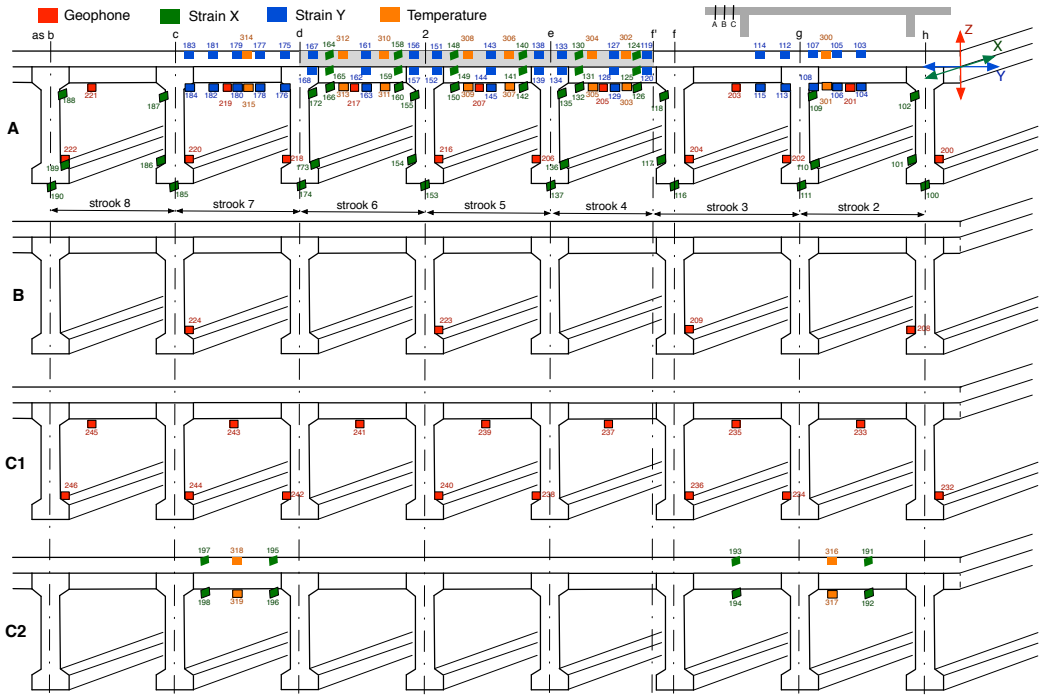


Figure 2.3: Diagram explaining the individual sensor placement on the Hollandse Brug.

quantitative estimate of the structural health of the Hollandse Brug.

- given the current sensor network deployment, obtain a new configuration of sensors, possibly employing fewer sensors, that is equivalent or comparable in terms of collected information.

In this thesis, we provide fundamental data mining methods and algorithms that can be employed to design a solution for the tasks above and related tasks involving sensors monitoring and analysis of systems.

Technical challenges

In order to make the goal of InfraWatch feasible from a data mining perspective, we need to identify a set of technical tasks which could serve, paired with domain knowledge in civil engineering, as basic tools to provide a solution. Moreover, as the bridge is affected by phenomena of different nature at

different temporal scales, the methods and solutions will have to cope with its multi-scale nature. The tasks we are interested in are described below:

- given a sensor-based time series, identify which are the relevant temporal scales of analysis.
- given a sensor-based time series, identify which are the recurring patterns in the data at multiple temporal scales.

In addition to the tasks above, we are also interested in visualizing effectively the large amount of time series data produced by the InfraWatch project.

Chapter 3

Preliminaries and Background

As discussed in the previous chapters, the primary focus of this thesis is the unsupervised analysis of sensor data collected from complex multi-scale systems. In this chapter, we review the main background concepts and tools that lay the foundations and prepare the discussion for the material in the remaining chapters.

We will start by introducing the concept of *convolution* and *signal filtering* [85]. As we deal with noisy measurements from real-world sensor networks, we will show how convolution and filtering can be used to reduce the effect of noise and support several other time series manipulation tasks. Building on the concept of convolution, we will then present a fundamental tool employed in the rest of the thesis that supports the analysis of a time series at multiple temporal scales: the *scale-space image* [103]. Finally, as our main focus is on the unsupervised modeling of phenomena in time series data, we will introduce the *Minimum Description Length principle* as our model selection framework of choice [34], motivate its adoption and discuss a simple application of it to noise removal.

3.1 Preliminaries

We start by giving some fundamental definitions used in this chapter and throughout the thesis. We deal with finite sequences of numerical measurements, collected by observing some property of a system with a sensor, and represented in the form of time series as defined below.

Definition (Time Series). A **time series** of length n is a finite sequence of values $\mathbf{x} = x[1], \ldots, x[n]$ of finite precision[1].

Throughout the thesis, we assume that the measurements are collected at uniformly spaced time points and according to a fixed sampling rate.

In many contexts, it is of particular interest to refer only to certain contiguous portions of a time series or subsequences, as defined below:

Definition (Subsequence). A subsequence $\mathbf{x}[a : b]$ of a time series $\mathbf{x}$ is defined as follows:

$$\mathbf{x}[a : b] = (\mathbf{x}[a], \mathbf{x}[a + 1], \ldots, \mathbf{x}[b]),\ a < b$$

3.2 Convolution and Filtering

Convolution is arguably one of the most important techniques in signal processing, for both the one-dimensional (time series) and two-dimensional case (images), and the fundamental operation in linear filtering. From a mathematical standpoint, convolution combines two functions to produce a third one, as defined below.

Definition (Convolution). Given two functions f and g, their **convolution $\mathbf{f} * \mathbf{g}$** is defined as the integral of their product after one of the functions is reversed and shifted:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)\, g(t - \tau)\, d\tau$$

[1] We assume 32-bit floating point values throughout the rest of this dissertation.

When referring to the process of filtering, the function f is said to be the signal to be filtered while the function g is called *convolution filter kernel*. Kernel functions directly define the properties of the filter. Specific kernel functions can be defined for amplification and attenuation, shifting or echoing of a signal. Other classes of kernel functions, as we will see shortly, can be used for low-pass and high-pass frequency filtering.

3.2.1 Convolution and LTI Systems

Although, in the context of this thesis, convolution is mainly used as a filtering operation, it is worth mentioning its deep connection with the theory of *linear time-invariant* (LTI) systems [85]. A system, defined by an input signal $x(t)$ and output signal $y(t)$, is said to be linear time-invariant if it satisfies the linearity and time-invariance properties. Linearity refers to the fact that a linear mapping exists between the inputs and the output of the system. More formally, given two inputs $x_1(t)$ and $x_2(t)$, respectively producing outputs $y_1(t)$ and $y_2(t)$, the scaled and summed input $a_1x_1(t) + a_2x_2(t)$ will produce $a_1y_1(t) + a_2y_2(t)$, where a_1 and a_2 are scalars. The property holds for any arbitrary number of terms.

On the other hand, time-invariance means that the output of the system does not depend on the particular time a given input is applied. In detail, given an input $x(t)$ and an output $y(t)$, the delayed input $x(t-\delta)$ will produce the delayed output $y(t-\delta)$.

Without diving into the specifics of the theory, we note that the operation of convolution fully describes the output of any arbitrary LTI system with a known impulse response. In fact, given an input signal $x(t)$ and an impulse response $h(t)$, the output of the associated LTI system is given by the convolution $x(t) * h(t)$. LTI systems and convolution play an important role in several technical fields such as signal processing, electronics, seismology [2], spectroscopy and control theory.

3.2.2 Discrete Convolution

Although we defined the convolution operation in the continuous domain, in practical cases, however, we deal with finite signals in the discrete domain (time series). The definition of convolution in the discrete case is presented below.

Definition (Discrete Convolution). Given a time series $\mathbf{x}$ of length n and a convolution filter kernel $\mathbf{h}$ of length m, the result of the **discrete convolution** $\mathbf{x} * \mathbf{h}$ is the time series $\mathbf{y}$ of length n, defined as:

$$\mathbf{y}[t] = \sum_{j=-m/2+1}^{m/2} \mathbf{x}[t-j]\,\mathbf{h}[j]$$

Note that since $\mathbf{x}$ is finite, $\mathbf{x}[t-j]$ may be undefined. To account for these boundary effects, $\mathbf{x}$ is padded with $m/2$ zeros before and after its defined range.

If not specified otherwise, from now on we will only refer to the discrete case of convolution.

3.2.3 Noise Filtering via Gaussian Smoothing

A common use of the convolution operator is smoothing a signal to remove noise or finer details. Smoothing can be obtained by employing several types of kernels like mean, median and Gaussian. In [12], the authors propose a methodology to mine interesting correlations in multivariate time series data based on generating features obtained by convoluting the input data with different kernels in order to enhance or reduce certain characteristic.

Gaussian smoothing, however, is a common noise filtering technique, as it cuts high frequencies, leaving untouched the low ones. It also has other useful properties as we will see in the next section.

Gaussian smoothing is based on the Gaussian kernel [75], as defined be-

low:

$$\mathbf{G}_\sigma = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

which, in the scope of this thesis, has a mean of 0, standard deviation σ and area under the curve equal to 1.

We can now define Gaussian smoothing as a particular convolution operation employing a Gaussian kernel.

Definition (Gaussian Smoothing). Given a time series $\mathbf{x}$ of length n and a Gaussian kernel $\mathbf{G}_\sigma$ discretized into m values, the result of the **Gaussian smoothing** is the time series $\mathbf{y}$ of length n, defined as:

$$\mathbf{x}[i] * \mathbf{G}_\sigma[i]$$

Note that to capture almost all non-zero values, we define $m = \lfloor 3\sigma \rfloor$.

The convolution acts as a *smoothing filter* which smooths each value $\mathbf{x}[t]$ based on its surrounding values. The amount of removed detail is directly proportional to the standard deviation σ (and thus m), from now on referred to as the *scale parameter*. In the limit, when $\sigma \to \infty$, the result of the Gaussian convolution converges to the mean of the signal $\mathbf{x}$ over the entire period involved.

To better picture the effect of Gaussian smoothing, consider the example in Figure 3.1. The top plot shows a signal collected from a strain sensor and the middle plot shows the same signal after being convoluted with a Gaussian kernel having $\sigma = 2$. The bottom plot highlights the part of the signal that has been removed by the filtering operation. Note how this residual signal does not just contain noise but it is still somewhat influenced by phenomena actually present in the signal, i.e. the peaks induced by passing vehicles. Designing the right noise removal filter ultimately boils down to choosing the right σ parameter for the given data.

In general, however, the choice of σ is strictly related to the task at hand and, consequently, to what is actually considered 'noise' in the given data. Consider, for example, a time series representing several months of strain measurements from a bridge deck at a high sampling rate (say 100 Hz or

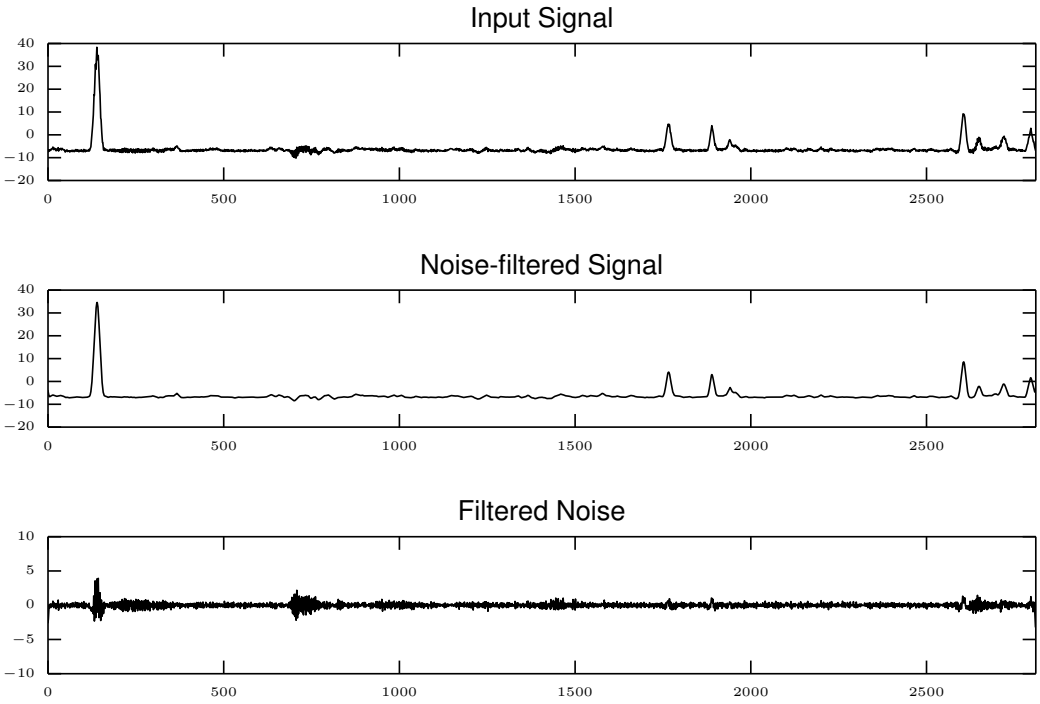


Figure 3.1: An example application of a Gaussian-based noise removal filter.

more). A typical bridge is affected by several phenomena at multiple temporal scales, ranging from events with a duration in the order of seconds such as passing cars and trucks, slightly longer ones such as congestion and weather conditions, to long-term ones like seasonal effects. The collected time series will represent all of these phenomena as a mixture. If we are interested in all the phenomena from the shortest to the longest, our concept of noise will coincide with anything lying below the temporal scale of the traffic events. On the other hand, if we are just interested in studying the effect of seasonal changes on the signal, we can ignore all the phenomena having shorter temporal scales and safely discard them as noise. This simple example illustrates how the concept of noise and the concept of scale of analysis are actually strictly interrelated and that a precise interpretation of noise can only be given by first looking at the task at hand.

In an unsupervised setting, it is not always clear what temporal scales we should look at and thus it is not always possible to determine in advance the

right value for σ. It follows from the definitions above that, by varying the parameter σ, Gaussian smoothing can be used to remove a fixed amount of detail from a signal. In other words, Gaussian smoothing can be interpreted as an operator that retains the information present above a certain temporal scale, where the scale is directly proportional to σ. This interpretation of Gaussian smoothing as scale parametrization is the core concept behind scale-space theory, a mathematical construction that we will use in the rest of the thesis and that we introduce in the next section.

3.3 Scale-Space Image

The *scale-space image* [103] is a scale parametrization technique for one-dimensional signals[2] based on convolution. Given a signal $\mathbf{x}$, the family of σ-smoothed signals $\Phi_{\mathbf{x}}$ over scale parameter σ is defined as follows:

$$\Phi_{\mathbf{x}}(\sigma) = \mathbf{x} * \mathbf{g}_{\sigma}\,,\ \sigma > 0$$

where $\mathbf{g}_{\sigma}$ is a Gaussian kernel having standard deviation σ, and $\Phi_{\mathbf{x}}(0) = \mathbf{x}$.

The signals in $\Phi_{\mathbf{x}}$ define a surface in the time-scale plane (t, σ) known in the literature as the *scale-space image* [62, 103]. This visualization gives a complete description of the scale properties of a signal in terms of Gaussian smoothing. Moreover, it has other properties useful for segmentation, as we will see in Section 4.3.1.

For practical purposes, the scale-space image is quantized across the scale dimension by computing the convolutions only for a finite number of scale parameters. More formally, for a given signal $\mathbf{x}$, we fix a set of scale parameters

$$S = \{2^i \mid 0 \leq i \leq \sigma_{max}\ \wedge i \in \mathbb{N}\}$$

and we compute $\Phi_{\mathbf{x}}(\sigma)$ only for $\sigma \in S$ where σ_{max} is such that $\Phi_{\mathbf{x}}(\sigma)$ is approximately equal to the mean signal of $\mathbf{x}$.

[2]From now on, we will use the term signal and time series interchangeably.

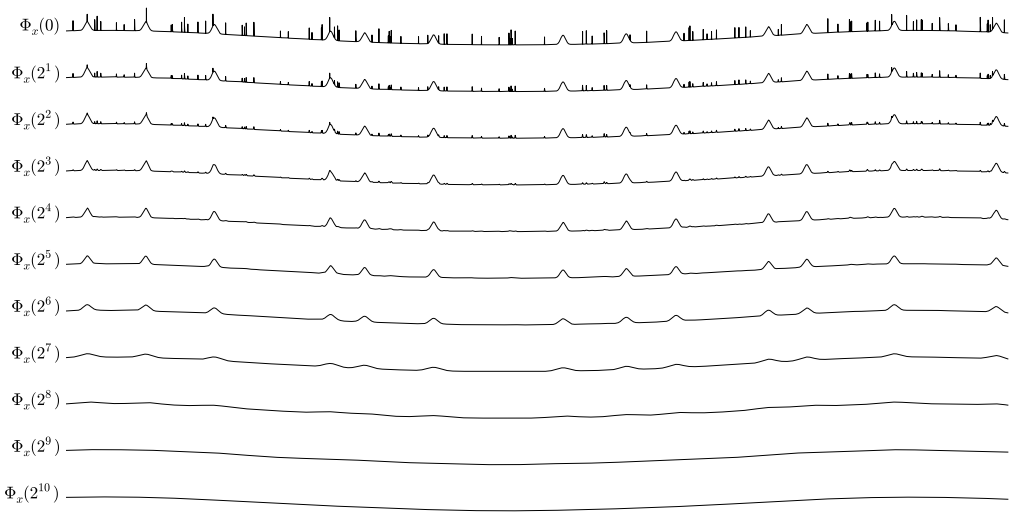


Figure 3.2: Scale-space image of an artificially generated signal totalling 259 200 points.

As an example, Figure 3.2 shows the scale-space image of an artificially generated signal. The top plot represents the original signal, constructed by three components at different temporal scales: a slowly changing and slightly curved baseline, medium-term events (bumps) and short-term events (peaks). It is easy to visually verify that, by increasing the scale parameter, a larger amount of detail is removed. In particular, the peaks are smoothed out at scales greater than $\sigma = 2^4$, and the bumps are smoothed out at scales greater than $\sigma = 2^8$, after which only the baseline remains.

3.3.1 Relation to the Zero-Crossings of Derivatives

The scale-space image has a number of interesting properties. An important one, that we will exploit throughout the thesis, is a property of Gaussian convolution that relates its zero-crossings, and those of its derivatives, to the scale parameter σ. In fact, as σ increases, the number of zero-crossings of the convoluted signal and of all its n-th derivatives can only remain constant or decrease [103]. Figure 3.3 demonstrates this concept in practice. The figure shows the relationship between the scale parameter σ and the number of zero-

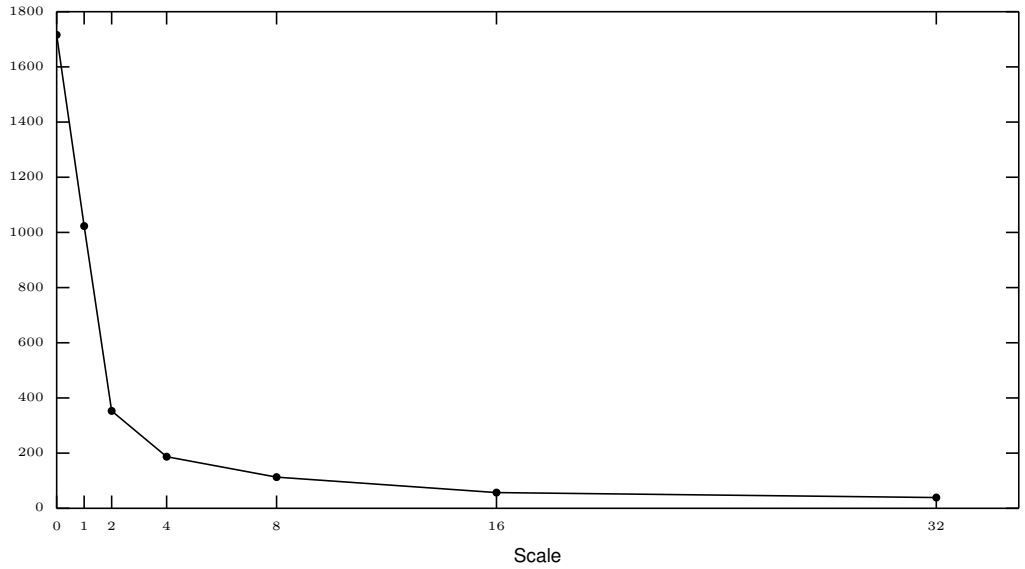


Figure 3.3: Relationship between the scale parameter and the number of zero-crossings of the first derivative of the signal shown in Figure 3.1.

crossing of the first derivative of the signal discussed in Figure 3.1.

3.4 Minimum Description Length

A recurring idea in this thesis is that learning and finding regularities in data can be seen as a form of *compression*. Compression is the act of representing some given data in the most compact way possible, such that its compressed representation has a lower number of bits than the original one [80]. The idea that the better we can compress a given data set, the more we can learn about it is a powerful one and it is formalized by the *Minimum Description Length principle*.

The Minimum Description Length [34] is an information-theoretic model selection framework that selects the best model according to its ability to *compress* the given data. In our context, the two-part MDL principle states that the best model M to describe the signal $\mathbf{x}$ is the one that minimizes the sum $L(M) + L(\mathbf{x} \mid M)$, where

- $L(M)$ is the length, in bits, of the description of the model,
- $L(\mathbf{x} \mid M)$ is the length, in bits, of the description of the signal when encoded with the help of the model M.

Given some data, MDL looks for a trade-off between the accuracy of a model and its complexity. Conceptually, MDL is a practical instantiation of the Occam's razor principle which states that, among several different hypotheses, the simplest is often also the best [91]. Moreover, MDL naturally protects against over-fitting as the principle takes into account the notion of model complexity and it discards models that are too complicated.

As we are dealing with unsupervised learning from time series data and we are interested in models that are as general as possible, the properties of the MDL framework makes it an ideal choice when designing model selection procedures.

In fact, prior work [42, 78, 88, 19, 10, 52] has already validated the effectiveness of the MDL approach when dealing with time series data and, throughout this thesis, we will further investigate its applicability to the analysis of sensor data.

3.4.1 Time Series Discretization

In order to use the MDL principle, we need to work with a quantized input signal and scale-space image. Because of this, we assume that the values v of the input signal $\mathbf{x}$ (and of the scale-space components $\Phi_{\mathbf{x}}(\sigma)$ for each considered σ) have been quantized to a finite number of symbols by employing the function defined below:

$$Q(v) = \left\lfloor \frac{v - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})}\, l \right\rfloor - \frac{l}{2}$$

where l, assumed to be even, is the number of bins to use in the discretization while $\min(\mathbf{x})$ and $\max(\mathbf{x})$ are respectively the minimum an maximum value in $\mathbf{x}$. Throughout the rest of the thesis, we assume $l = 256$.

One question that might arise is if such a quantization removes meaningful information from the time series. In [42], the authors show that the effect of quantization is rather modest on several time series from various domains.

3.4.2 MDL Noise Filtering

In 3.2.3, we have shown how Gaussian convolution can be used to remove high-frequency components from a given signal and, thus, serve as a noise removal filter. We stressed, however, that the choice of the parameter σ, is a critical one and strictly depends on the characteristics of the data at hand. In this section, we use the Minimum Description Length principle to select the optimal σ given a time series, where by *optimal* we mean the one that retains the most characteristic information in the data.

Assume we are given a time series $\mathbf{x}$ of length n and, as we are using MDL, its values have been discretized to a fixed cardinality (in this example, 256 possible values) using the quantization function Q introduced in above.

In order to frame the problem from an MDL perspective, we first have to define what the possible models for $\mathbf{x}$ are. We consider the components of the scale-space $\Phi_{\mathbf{x}}$, quantized to different cardinalities, as models. In other words, given a scale parameter σ and a cardinality c, we define a model for $\mathbf{x}$ as $M_{\sigma,c} = Q(\Phi_{\mathbf{x}}(\sigma), c)$, where Q is a quantization function.

The MDL principle states that the best model to compress $\mathbf{x}$ minimizes the sum $L(M) + L(\mathbf{x} \mid M)$. We define the description length of a model in terms of its entropy as $nH(\mathbf{x})$, where H is the entropy function. The description length of x when encoded with the help of a model refers to the complexity of the residual $nH(\mathbf{x}-M)$, that is the information discarded by the model.

Figure 3.4 shows the results of the approach that we just discussed in a practical scenario. The top plot depicts the input time series: a detail of a peak in a bridge's strain sensor signal caused by a passing vehicle. The plot in the middle shows the time series after being Gaussian-smoothed with the optimal σ, selected as discussed. Note how both the overall shape of the

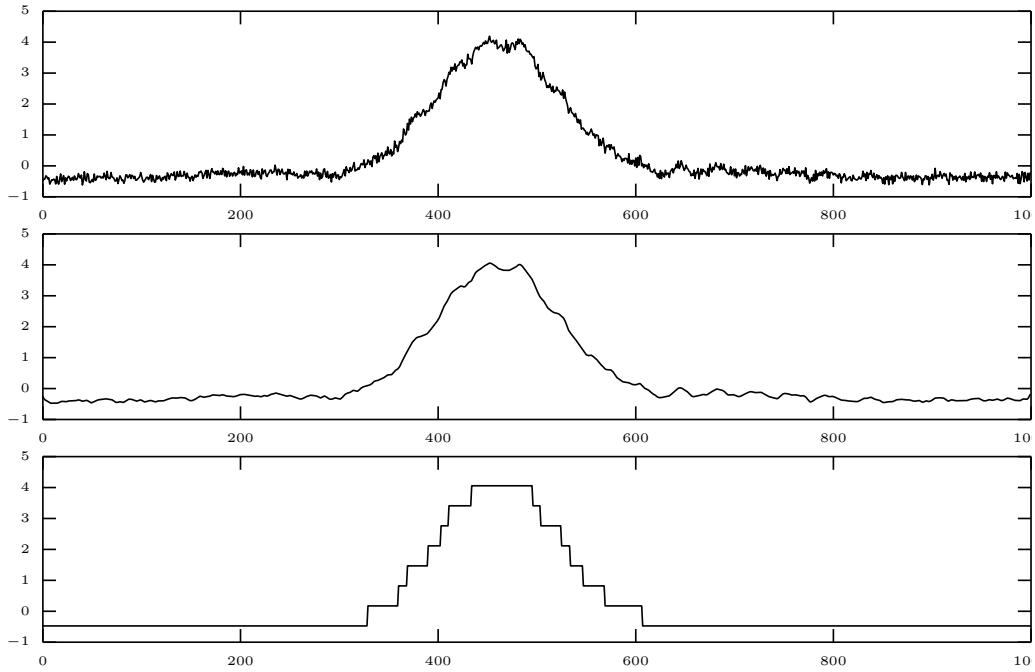


Figure 3.4: Example of MDL-based noise filtering.

signal and the subtle vibrations iduced by the passing truck are retained. The bottom plot, finally, shows the optimal model (quantized) according to MDL. For this particular example, we considered $\sigma \in \{2, 2^2, 2^3, 2^4, 2^5, 2^6\}$ and the cardinality $c \in \{4, 8, 16, 32, 64, 128, 256\}$. The optimal model is given by $\sigma = 2^3$ and $c = 8$.

A similar approach to noise removal has been taken by Miao et al. [71] as a preprocessing step in the context of pattern detection in time series data.

Chapter 4

Identifying the Relevant Temporal Scales

4.1 Introduction

When monitoring complex physical systems over time, one often finds multiple phenomena in the data that work on different time scales. If one is interested in analyzing and modeling these individual phenomena, it is crucial to recognize these different scales and separate the data into its underlying components. In this chapter, we present a method for extracting the time scales of various phenomena present in large time series. The method combines concepts from the signal processing domain with feature selection and the Minimum Description Length principle [34] which we introduced in Chapter 3.

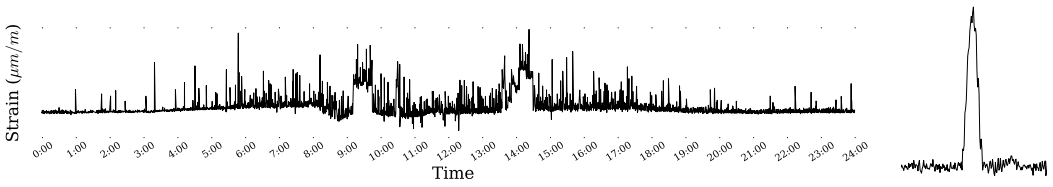
We introduced the need for analyzing time series data at multiple time scales in Chapter 1 and we discussed in Chapter 2 how this is nicely demonstrated by the InfraWatch project.

In this project, we employ a range of sensors to measure the dynamic response of the Hollandse Brug, a large Dutch highway bridge to varying traffic and weather conditions. When viewing this data (see Fig. 4.1a), one can easily distinguish various *transient events* in the signal that occur on different time

scales. Most notable are the gradual change in strain over the course of the day (as a function of the outside temperature, which influences stiffness parameters of the concrete), a prolonged increase in strain caused by rush hour traffic congestion, and individual bumps in the signal due to cars and trucks traveling over the bridge. In order to understand the various changes in the sensor signal, one would benefit substantially from separating out the events at various scales. The main goal of the work described here is to do just that: we consider the temporal data as a series of superimposed effects at different time scales, establish at which scales events most often occur, and from this we extract the underlying signal components.

We approach the scale selection problem from a Minimum Description Length (MDL) perspective (see Section 3.4). The motivation for this is that we need a framework in which we can deal with a wide variety of representations for scale components. The MDL framework was shown to be sufficiently general to provide this flexibility by Hu et al. [42] for the problem of choosing the best model for a given signal. Our main assumption here is that separating the original signal into components at different time scales will simplify the shape of the individual components, making it easier to model them separately. Our results show that, indeed, these multiple models outperform (in terms of MDL score) a single model derived from the original signal. While introducing multiple models incurs the penalty of having to describe these multiple models, there are much fewer 'exceptions' to be described compared to the single model, yielding a lower overall description length. For instance, in the sensor data of Fig. 4.1a, cars are often passing in one direction while there is rush hour congestion in the opposite direction. Using multiple models, this is modeled accurately, while a single model will easily ignore these events.

As we discussed in detail in Section 3.3, the analysis of time scales in time series data is often approached from a *scale-space* perspective, which involves convolution of the original signal with Gaussian kernels of increasing size [103] to remove information at smaller scales. By subtracting carefully selected components of the scale-space, we can effectively cut up the scale space into k ranges. In other words, signal processing offers methods for producing a large



(a) (b)

Figure 4.1: (a) One day of strain measurements from a large highway bridge in the Netherlands. The multiple external factors affecting the bridge are visible at different time scales. (b) A detail of plot (a) showing one of the peaks caused by passing vehicles.

collection of derived features, and the challenge we face in this chapter is how to select a subset of k features, such that the original signal is decomposed into a set of meaningful components at different scales.

Our approach applies the MDL philosophy to various aspects of modeling: choosing the appropriate scales at which to model the components, determining the optimal number of components (while avoiding overfitting on overly specific details of the data), and deciding which class of models to apply to each individual component. For this last decision, we propose two classes of models representing the components respectively on the basis of a discretization and a segmentation scheme. For this last scheme, we allow three levels of complexity to approximate the segments: piecewise constant approximations, piecewise linear approximations, as well as quadratic ones. These options result in different trade-offs between model cost and accuracy, depending on the type of signal we are dealing with.

A useful side product of our approach is that it identifies a concise representation of the original signal. This representation is useful in itself: queries run on the decomposed signal may be answered more quickly than when run on the original data. Furthermore, the parameters of the encoding may indicate useful properties of the data as well.

The rest of the chapter is organized as follows. Section 4.2 introduces the concept of scale-space decomposition. Section 4.3 shows how we encode

the signal decompositions and use MDL to select the best subset of scales. Section 4.4 presents an empirical evaluation of our method on both real-world and artificial data. Section 4.5 links our method to related work. Finally, Section 4.6 states our main conclusions and ideas for future work.

4.2 Scale-Space Decomposition

In this section, we show how to manipulate the scale-space image to filter out the effects of transient events in a specific range of scales. This will lead to the definition of a signal decomposition scheme.

Along the scale dimension of the scale-space image, short-time transient events in the signal will be smoothed away sooner than longer ones. In other words, we can associate with each event a maximum scale σ_{cut} such that, for $\sigma > \sigma_{cut}$, the transient event is no longer present in $\Phi_{\mathbf{x}}(\sigma_{cut})$. This fact leads to the following two observations:

- Given a signal scale-space image $\Phi_{\mathbf{x}}$, the signal $\Phi_{\mathbf{x}}(\sigma)$ is only affected by the transient events at scales greater than σ. This is conceptually equivalent to a *low-pass filter* in signal processing.
- Given a signal scale-space image $\Phi_{\mathbf{x}}$ and two scales $\sigma_1 < \sigma_2$, the signal $\Phi_{\mathbf{x}}(\sigma_1) - \Phi_{\mathbf{x}}(\sigma_2)$ is mostly affected by those transient events present in the range of scales (σ_1, σ_2). This is similar to a *band-pass filter* in signal processing.

As an example, reconsider the signal $\mathbf{x}$ and its scale-space image $\Phi_{\mathbf{x}}$ of Figure 3.2. Figure 4.2 shows (from top to bottom):

- the signal $\Phi_{\mathbf{x}}(0) - \Phi_{\mathbf{x}}(2^4)$, which is the result of a high-pass filtering; this feature represents the short-term events (peaks),
- the signal $\Phi_{\mathbf{x}}(2^4) - \Phi_{\mathbf{x}}(2^{10})$, which is the result of a band-pass filtering; this feature represents the medium-term events (bumps),
- the signal $\Phi_{\mathbf{x}}(2^{10})$, which is the result of a low-pass filtering; this feature represents the long-term trend.

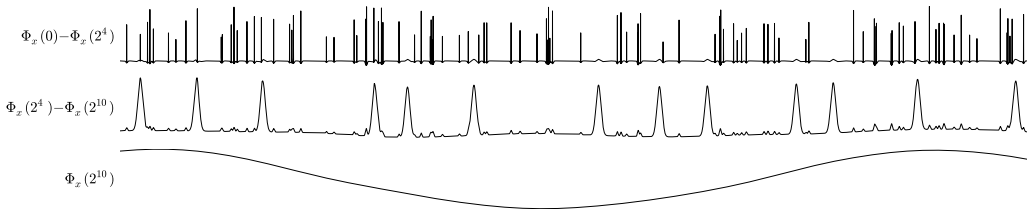


Figure 4.2: Examples of signal decomposition obtained from the scale-space image in Figure 3.2.

Generalizing the example in Figure 4.2, we can define a decomposition scheme of a signal $\mathbf{x}$ by considering adjacent ranges of scales of the signal scale-space image. We formalize this idea below.

Definition (Scale-Space Decomposition). Given a signal $\mathbf{x}$ and a set of $k-1$ scale parameters $C = \{\sigma_1, \ldots, \sigma_{k-1}\}$ (called the cut-points set) such that $\sigma_1 < \ldots < \sigma_{k-1}$, the **scale decomposition** of $\mathbf{x}$ is given by the set of component signals $D_{\mathbf{x}}(C) = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$, defined as follows:

$$\mathbf{x}_i = \begin{cases} \Phi_{\mathbf{x}}(0) - \Phi_{\mathbf{x}}(\sigma_1) & \text{if } i = 1 \\ \Phi_{\mathbf{x}}(\sigma_{i-1}) - \Phi_{\mathbf{x}}(\sigma_i) & \text{if } 1 < i < k \\ \Phi_{\mathbf{x}}(\sigma_{k-1}) & \text{if } i = k \end{cases}$$

Note that for k components we require $k-1$ cut-points. This decomposition has several elegant properties:

- $\mathbf{x}_k$ can be seen as the baseline of the signal, as obtained by a low-pass filter;
- $\mathbf{x}_i$ for $1 \leq i < k$ are signals as obtained by a band-pass filter, and can be used to identify transient events;
- $\sum_{i=1}^{k} \mathbf{x}_i = \mathbf{x}$, i.e., the original signal can be recovered from the decomposition.

4.3 MDL Scale Decomposition Selection

Given an input signal $\mathbf{x}$, the main computational challenge we face is twofold:

- find a good subset of cut-points C such that the resulting k components of the decomposition $D_{\mathbf{x}}(C)$ optimally capture the effect of transient events at different scales,
- select a representation for each component, according to its inherent complexity.

As stated before, the rationale behind the scale decomposition is that it is easier to model the effect of a single class of transient events at a given scale than to model the superimposition of many, interacting transient events at multiple scales. We thus need to trade off the added complexity of having to represent multiple components for the complexity of the representations themselves.

We approach this model selection problem by using the Minimum Description Length (MDL) principle introduced in Section 3.4. The possible candidate models depend on the scale decomposition $D_{\mathbf{x}}(C)$ considered[1] and on the representations used for its individual components. An ideal set of representations would adapt to the specific features of every single component, resulting in a concise summarization of the decomposition and, thus, of the signal. In order to apply the MDL principle, we need to define a model $M_{D_{\mathbf{x}}(C)}$ for a given scale decomposition $D_{\mathbf{x}}(C)$ and, consequently, how to compute both $L(M_{D_{\mathbf{x}}(C)})$ and $L(\mathbf{x} \mid M_{D_{\mathbf{x}}(C)})$. The latter term is the length in bits of the information lost by the model, i.e., the residual signal $\mathbf{x} - M_{D_{\mathbf{x}}(C)}$.

As the MDL framework is only applicable to discrete data, we assume that the input signal $\mathbf{x}$ and the results of all the subsequent operations are discretized as discussed in Section 3.4.1. In the next sections, we introduce the proposed representation schemes for the components and define the bit complexity of the residual and the model selection procedure.

[1] Including the decomposition formed by zero cut-points ($C = \emptyset$), i.e., the signal itself.

4.3.1 Component Representation Schemes

Within our general framework, many different approaches could be used for representing the components of a decomposition. In the next paragraphs we introduce two such methods.

Discretization-based representation

In some components of our data transient events always occur with similar amplitudes, mixed with long stretches of baseline values (see Figure 4.2). Hence, a desirable encoding could be one that captures this repetitiveness in the data by giving short codes to long stretches of the baseline and the commonly occurring amplitudes. Unfortunately, our original discretization is too fine-grained to capture regular occurrences of similar amplitudes. As a first representation, we hence propose to also consider more coarse-grained discretizations of the original range of values. We do this by discretizing each value v in a component to a value $\lfloor Q(v)/2^i \rfloor$, where several values for i are considered for each component, typically $i \in \{2, 4, 6\}$. By doing so, similar values will be grouped together in the same bin. The resulting sequence of integers is compacted further by performing run-length encoding, resulting in a string of (v, l) pairs, where l represents the number of times value v is repeated consecutively. This string is finally encoded using a Shannon-Fano or Huffman code (see Section 4.3.2).

As a simplified illustration of how the MDL principle helps here to identify components, consider data generated by the expression $(67)^n(01)^n$ ($4n$ integers from the range $\{0, \ldots, 2^3 - 1\}$), where we assume n and the range are fixed. In this data, each symbol occurs with the same frequency; we can encode the time series hence with $-\log_2(1/4) \cdot 4 \cdot n = 8n$ bits for the data, plus $8 \log n$ bits for the dictionary of frequencies. Consider now the decomposition of the signal into two time series, $6^{2n}0^{2n}$ and $(01)^{2n}$. The first component, of which the run-length encoding is $(6, 2n)(0, 2n)$, can be encoded using only 2 bits for the time series (as there is only one possible run-length value, we use 0 bits to encode the run-lengths), $8 \log n$ bits for the dictionary of ampli-

tudes, and $3 \log n$ bits to identify the length of the one run-length ($\log n$ bit for identifying the number of run-lengths, in this case one, $\log n$ to identify the one run-length present, and $\log n$ to identify its frequency, from which the encoding with 0 bits follows). The second component can be encoded using $4n$ bits for the time series, as well as $8 \log n$ bits for the dictionary. Assuming we also use 1 bit per component to identify the type of encoding used, this gives us an encoding in $4 + 19 \log n + 4n$ bits. Comparing this to $8n + 8 \log n$ bits, for $n \geq 11$ we will hence correctly identify the two components in this simplified data.

Segmentation-based representation

The main assumption on which we base this method is that a clear transient event can be accurately represented by a simple function, such as a polynomial of a bounded degree. Hence, if a signal contains a number of clear transient events, it should be possible to accurately represent this signal with a number of segments, each of which represented by a simple function.

Given a component $\mathbf{x}_i$ of length n, let

$$z(\mathbf{x}_i) = \{t_1, t_2, \ldots, t_m\}, \quad 1 < t_i \leq n$$

be a set of indexes of the segment boundaries.

Let $\texttt{fit}(\mathbf{x}_i[a : b], d_i)$ be the approximation of $\mathbf{x}_i[a : b]$ obtained by fitting a polynomial of degree d_i. Then, we represent each component $\mathbf{x}_i$ with the approximation $\hat{\mathbf{x}}_i$, such that:

$$\begin{aligned}
\hat{\mathbf{x}}_i[0 : z_1] &= \texttt{fit}(\mathbf{x}_i[0 : z_1], d_i) \\
\hat{\mathbf{x}}_i[z_i : z_{i+1}] &= \texttt{fit}(\mathbf{x}_i[z_i : z_{i+1}], d_i), \quad 1 \leq i < m \\
\hat{\mathbf{x}}_i[z_m : n] &= \texttt{fit}(\mathbf{x}_i[z_m : n], d_i)
\end{aligned}$$

Note that approximation $\hat{\mathbf{x}}_i$ is quantized again by reapplying the function Q to each of its values.

For a given k-components scale decomposition $D_{\mathbf{x}}(C)$ and a fixed polynomial degree for each of its components, we calculate the complexity in bits

of the model $M_{D_{\mathbf{x}}(C)}$, based on this representation scheme, as follows. Each approximated component $\hat{\mathbf{x}}_i$ consists of $|z(\mathbf{x}_i)| + 1$ segments. For each segment, we need to represent its length and the $d_i + 1$ coefficients of the fitted polynomial. The length ls_i of the longest segment in $\hat{\mathbf{x}}_i$ is given by

$$ls_i = \max(z_1 \cup \{z_{i+1} - z_i \mid 0 < i \leq m\})$$

We therefore use $\log_2(ls_i)$ bits to represent the segment lengths, while for the coefficients of the polynomials we employ floating point numbers of fixed[2] bit complexity c. The MDL model cost is thus defined as:

$$L(M_{D_{\mathbf{x}}(C)}) = \sum_{i=1}^{k} (|z(\mathbf{x}_i)| + 1) \left(\lceil \log_2(ls_i) \rceil + c\,(d_i + 1)\right)$$

So far we assumed to have a set of boundaries $z(\mathbf{x}_i)$, but we did not specify how to compute them. A desirable property for our segmentation would be that a segmentation at a coarser scale does not contain more segments than a segmentation at a finer scale. The scale space theory assures that there are fewer zero-crossing of the derivatives of a signal at coarser scales [103]. In our segmentation, we use the zero-crossings of the first and second derivatives. More formally, we define the segmentation boundaries of a component $\mathbf{x}_i$ to be

$$z(\mathbf{x}_i) = \left\{ t \in \mathbb{R} \;\middle|\; \frac{d\mathbf{x}_i}{dt}(t) = 0 \right\} \bigcup \left\{ t \in \mathbb{R} \;\middle|\; \frac{d^2\mathbf{x}_i}{dt}(t) = 0 \right\}.$$

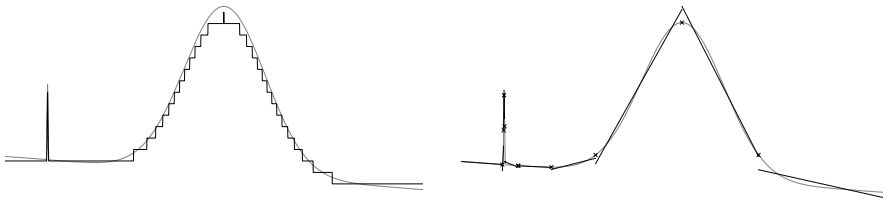
Figure 4.3b shows an example of segmentation obtained as above using fitted polynomials of degree 1.

However, many other segmentation algorithms are known in the literature [48, 53] and all of them can be interchangeably employed in this context.

4.3.2 Residual Encoding

Given a model $M_{D_{\mathbf{x}}(C)}$, its residual $\mathbf{r} = \mathbf{x} - \sum_{i=1}^{k} \hat{\mathbf{x}}_i$, computed over the components approximations, represents the information of $\mathbf{x}$ not captured

[2] In our experiments $c = 32$.



(a) (b)

Figure 4.3: Example of discretization-based encoding (a) and segmentation-based encoding with first degree polynomial approximations (the markers show the zero-crossings) (b).

by the model. Having already defined the model cost for the two proposed encoding schemes, we only still need to define $L(\mathbf{x} \mid M_{D_{\mathbf{x}}(C)})$, i.e., a bit complexity $L(\mathbf{r})$ for the residual $\mathbf{r}$.

Here, we exploit the fact that we operate in a quantized space; we encode each bin in the quantized space with a code that uses approximately $-\log(P(x))$ bits, where $P(x)$ is the frequency of the xth bin in our data. The main justification for this encoding is that we expect that the errors are normally distributed around 0. Hence, the bins in the discretization that reflect a low error will have the highest frequency of occurrences; we will give these the shortest codes. In practice, such codes can be obtained by means of Shannon-Fano coding or Huffman coding; as Hu et al. [42] we use Huffman coding in our experiments.

4.3.3 Model Selection

We can now define the MDL score that we are optimizing as follows:

Definition (MDL Score). Given a model $M_{D_{\mathbf{x}}(C)}$, its **MDL score** is defined as:

$$L(M_{D_{\mathbf{x}}(C)}) + L(\mathbf{r})$$

In the case of discretization-based encoding, the MDL score is affected by the cardinality used to encode each component. In the case of segmentation-

based encoding the MDL score depends on the boundaries of the segments and the degrees of the polynomials in the representation. In both cases, also the cut-points of the considered decomposition affect the final score.

The simplest way to find the model that minimizes this score is to enumerate, encode and compute the MDL score for every possible scale-space decomposition and all possible encoding parameters. As we shall now show, this brute-force approach is practically feasible.

The number of possible scale decompositions depends on the total number of cut-points sets we can build from the computed scale parameters in $\Phi_{\mathbf{x}}$. We fix the maximum number of cut-points in a candidate set to some value c_{max}. This also means that we limit our search to those scale decompositions having $c_{max} + 1$ components or less. Moreover, given our wish to consider only simple approximations of the signals, we can also assume a reasonably low limit d_{max} (in practice, $d_{max} = 2$) on the degree of the polynomials that approximate the segments of each given component.

Computing the MDL score for each encoded scale decomposition, obtained by ranging over all the possible configurations of cut-points $C_1, \ldots, C_{k-1}$, and all the possible configurations of polynomial degrees $d_1, \ldots, d_k$, hence requires calculating MDL scores for

$$\sum_{k=2}^{c_{max}+1} \binom{|\mathbf{S}|}{k-1} d_{max}^k$$

scale decompositions. This turns out to be a reasonable number in most practical cases we consider, and hence we use an exhaustive approach in our experiments.

4.4 Experiments

In this section, we experimentally evaluate our method, both on artificial data and on actual sensor data from the highway bridge mentioned in the introduction. To evaluate the strengths and weaknesses of our method, we have

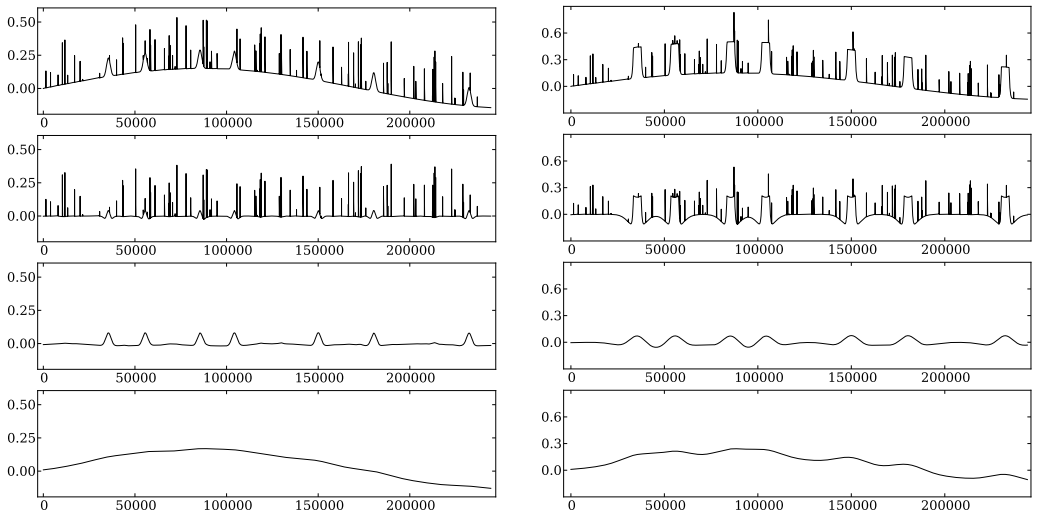


Figure 4.4: Signals (top) and top-ranked decompositions for the two artificial datasets.

tested it on a range of artificial datasets[3] that mimic some of the multi-scale phenomena present in the bridge data. Our constructed data deliberately varies from easy, with clearly separated scales, to challenging with a variety of event shapes and sizes. All artificial datasets represent sensor data measured at 1 Hz for a duration of three days (totaling 259,200 data points). The data was produced by combining three components at three distinct scales, resembling 1) individual events from vehicles, 2) traffic jams that last several tens of minutes, and 3) gradual change of the baseline, due to temperature changes of the bridge over the course of several days.

Artificial data

We start by considering one particular dataset in detail (see Figure 4.4a). This dataset was constructed by using Gaussian shapes for both the small and medium-scale events, and a sine wave of period 2.25 days at the largest

[3]The artificial datasets and the source code can be obtained by contacting the first author.

scale. Medium events have a constant height, whereas small-scale events have a random height. We limited the search space to decompositions having a maximum of 4 components (3 cut-points). As can be seen in Figure 4.4a, our method was able to identify the fact that this data contains three important scales. Furthermore, the method correctly identified the two necessary cut-points, such that the three original components were reconstructed. The selected cut-points[4] appear at scales $2^9 = 512$ and $2^{12} = 4096$. When considering the separated components in detail, some influence across the scale-boundaries is visible, for example where small effects of the 'traffic jams' appear among the small-scale events. These effects seem unavoidable, with the inherent limitations of the scale-space-based band-pass filtering and the discrete collection of scales we consider (powers of 2).

This optimal result has an MDL-score of 509,000 bits, being the sum of the model cost ($L(M) = 75,072$) and the error length ($L(D \mid M) = 433,928$). The second-ranked result on this data, with cut-points $C = \{2^{11}, 2^{13}\}$, shows a similar result, however with slightly more pronounced cross-boundary artifacts in the smallest scale, as is expected with a doubling of the lower cut-point. The MDL-score of this result is $64,896 + 450,487 = 515,383$. The $k = 1$ case, which corresponds to compression of the original signal without any decomposition, appears at rank three, with an MDL-score of $44,640 + 471,271 = 515,911$. This model obviously has a much lower model cost, due to having to represent only a single component, but this is compensated by the substantially higher error length, putting it below the scale-separated results. Ranks four and five represent two $k = 2$ results, where the former groups the small and medium scales together, and the latter the medium and large. All results in the top 10 relate to models that use polynomial representations ($d \leq 2$).

Not all artificial datasets considered produced perfect results. In Figure 4.4b, we show an example of a dataset that includes 'traffic jams' that resemble more closely some of the phenomena in the actual sensor data. In many cases, traffic jams appear fairly rapidly, and then show an increased load on

[4]Note that our method returns the boundaries between scales, rather than the actual scales of the original components.

the bridge over a prolonged period. This is modeled in the data by medium-scale events that start and stop fairly rapidly, and remain constant in the meantime. The best result found, with cut-points $C = \{2^{12}, 2^{13}\}$, is shown in Figure 4.4b. This demonstrates that the proposed method is not able to properly separate the medium and low-scale events. In fact, even though the medium component does identify the location of the 'traffic jams', most of the rectangular nature is accounted for by the small scale. To some extent, this is understandable, as the start and end of the event could be considered high-frequency events with rapid changes in value. Therefore, parts of these events appear at a small scale, and the algorithm is mirroring this effect. In any case, the algorithm *is* able to identify the correct number of components, and is able to produce indications as to the location of the traffic jams. The top four results all show similar mixtures of scales, whereas the rank-five result groups the lowest two scales together. The $k = 1$ result appears at rank 14.

In order to better understand to what extent the proposed method is able to separate components at different scales, we carried out a more controlled experiment. We generated 11 different datasets constructed from 3 components. We fixed the scales of the short-term and long-term components respectively around $\sigma = 2^3$ and $\sigma = 2^{15}$, while the scale of the medium-term component varies from dataset to dataset in the range $(2^4, \ldots, 2^{14})$. The table below shows the number of components (k) of the top-ranked decomposition for the 11 datasets according to the scale parameter σ of the medium-term component.

σ	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}
k	1	2	2	2	3	3	3	3	1	1	1

As the table suggests, the proposed method fails to identify the right number of components when the scales are too close to each other. However, when the scales are separated sufficiently ($2^8 \leq \sigma \leq 2^{11}$), the right number of components is identified. Also in this case, all the top-ranked decompositions relate to models that use polynomial representations.

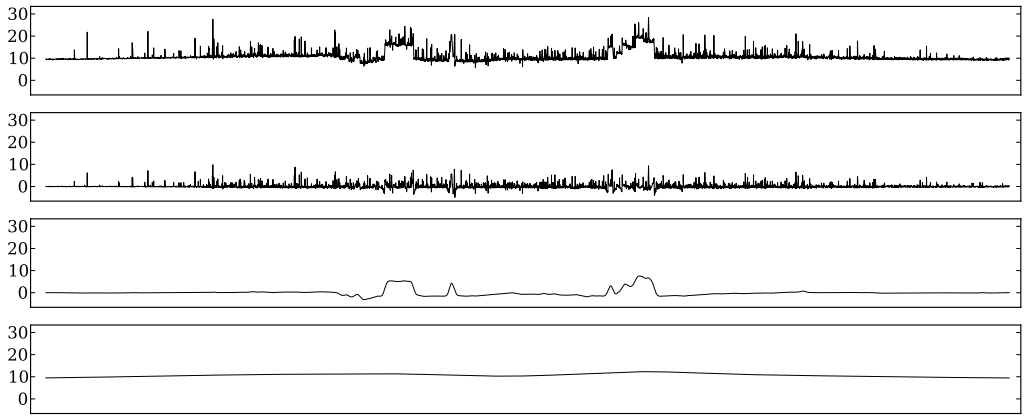


Figure 4.5: Signal (top) and top-ranked scale decomposition for the InfraWatch data.

InfraWatch data

As anticipated by the motivating example in the introduction, we consider the strain measurements produced by a sensors attached to a large highway bridge in the Netherlands. For this purpose, we consider a time series consisting of 24 hours of strain measurements sampled at 1 Hz (totaling $86,400$ data points). A plot of the data is shown in Figure 4.5 (topmost plot). We evaluated all the possible decompositions up to three components (two cut-points) allowing both the representation schemes we introduced. In the case of the discretization-based representations, we limit the possible cardinalities to 4, 16 and 64.

The top-ranked decomposition results in 3 components as shown in the last three plots in Figure 4.5. The selected cut-points appear at scales $2^6 = 64$ and $2^{11} = 2048$. All three components are represented with the discretization-based scheme, with a cardinality of respectively 4, 16, and 16 symbols. The decomposition has an MDL-score of $344,276$, where $L(M) = 19,457$ and $L(D \mid M) = 324,818$. The found components accurately correspond to physical events on the bridge. The first component, covering scales lower than 2^6, reflects the short-term influence caused by passing vehicles and represented as peaks in the signal. Note that the cardinality selected for this component is the lowest admissible in our setting (4). This is reasonable considering that

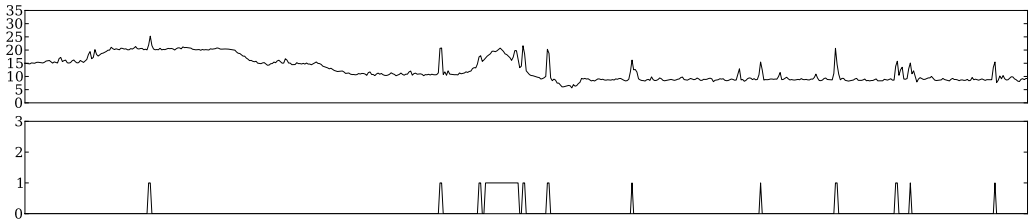


Figure 4.6: A detail of the original strain signal (one hour) and the selected first component as represented with 4 symbols.

the relatively simple dynamic behavior occurring at these scales, mostly the presence or not of a peak over a flat baseline, can be cheaply described with 4 or fewer states without incurring a too large error. The middle component, covering scales between 2^6 and 2^{11}, reflects the medium-term effects caused by traffic jams. As in the artificial data, the first component is slightly influenced by the second one, especially at the start and ending points of a traffic jam. Finally, the third component captures all the scales greater than 2^{11}, here representing the effect of temperature during a whole day. To sum up, the top-ranked decomposition successfully reflects the real physical phenomena affecting the data. The decompositions with rank 8 or less all present similar configurations of cut-points and cardinalities, resulting in comparable components where the conclusions above still hold. The first 2-component decomposition appears at rank 10 with the cut-point placed at scale 2^6, which separates the short-term peaks from all the rest of the signal (traffic jams and baseline mixed together). These facts make the result pretty stable as most of the good decompositions are ranked first.

Dike monitoring data

We evaluated the method on an additional real-world scenario: the sensor-based monitoring of dikes [20]. We focus on the data produced by a network of pore sensors installed on a sea dike in Boston (UK). The considered data consists of one time series, representing one year of measurements sampled every 15 minutes (for a total of 27 610 values), ranging from October 2011 to October 2012. As many complex physical systems, also dikes are affected

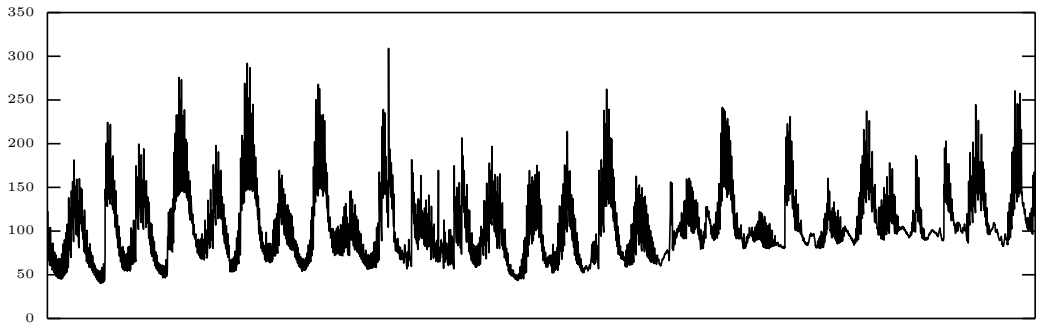


Figure 4.7: One year of pore pressure measurements from the one of the sensors installed on a sea dike in Boston (UK).

by multiple phenomena at different temporal scales and the effect is visible in the produced sensor data. Figure 4.7 shows one year of measurements from one of the pore sensors in the network. The sensor exhibits two clear periodic trends at different temporal scales. At the shortest temporal scale, the water level follows half-daily tides (see the detailed plots of Figure 4.7 and Figure 4.8). The water level, however, is also affected, on a two-weekly basis, by the lunar cycles that have a periodic effect on the overall amplitude of the signal.

For the considered time-series, we evaluated all the possible decompositions up to three components (two cut-points) employing the discretization-based scheme with possible cardinalities limited to 4, 8, and 16.

The top-ranked decomposition for the time series is shown in Figure 4.9. The selected cut-points appear at scales $2^5 = 32$ and $2^{13} = 8192$. All three components, represented with the discretization-based scheme, have a selected cardinality of 8 symbols. The decomposition has an MDL-score of $213,288$, where $L(M) = 19,075$ and $L(D \mid M) = 194,213$. Note that the first two components, from top to bottom, effectively separate the two periodic trends in the data, i.e. the half-daily tidal effects (a detail of which can be seen in Figure 4.10) and the two-weekly cycles due to lunar tides. The third long-term component reflects slow change happening in the dike due to the effect of humidity. However, the trend also includes a slight change in the sensor sensitivity itself, as its response is affected by the external temperature.

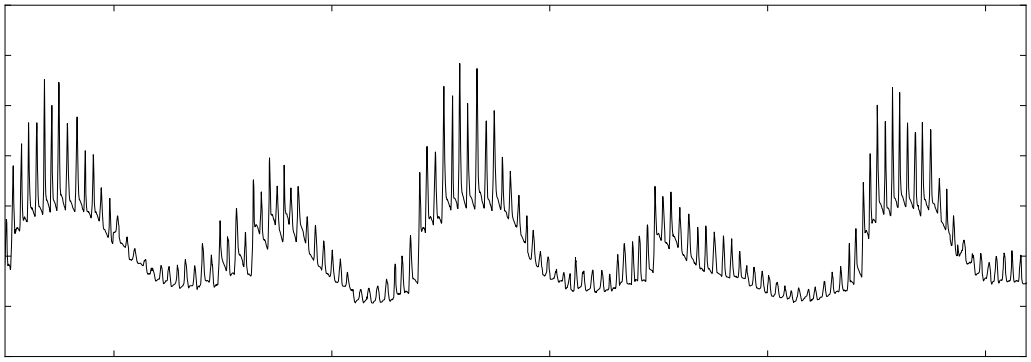


Figure 4.8: Zoomed detail of Figure 4.7. The figure clearly shows the presence of short term periodic trends, due to the half-daily effect of the tides.

An application: detecting passing vehicles

The component selection and representation generated by the MDL procedure may be useful in itself for tasks such as classification. For example, consider the short-term component of the previous example, Figure 4.5 (second plot). It represents the traffic activity over the bridge and has been represented with a discretization-based scheme using 4 symbols. Figure 4.6 shows a detail (1 hour) of the discretized component (bottom) and the relative original signal (top). The first 2 symbols (0 and 1) respectively classify the absence or presence of a passing vehicle, while the other two, considerably less frequent, are outliers in the data. The represented component, as selected by MDL, can thus be used to monitor traffic activity over the bridge, a task that is considerably more challenging using the original signal, due to the variations introduced by temperature fluctuations and traffic jams.

4.5 Related Work

Papadimitriou et al. [76] propose a method to discover the key trends in a time series at multiple time scales (window lengths) by defining an incremental version of Singular Value Decomposition. In signal processing, Indepen-

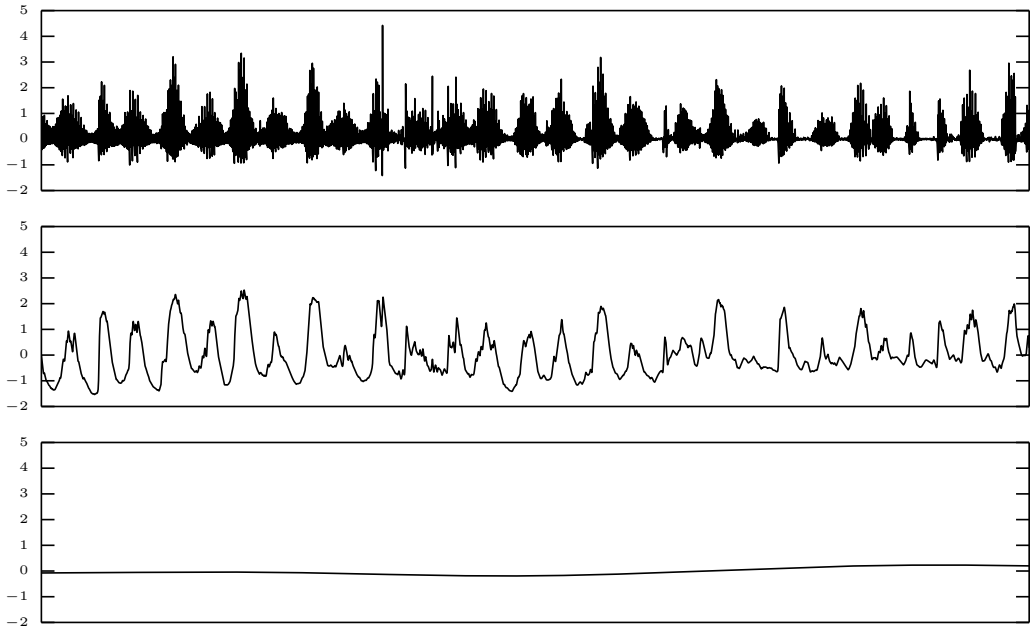


Figure 4.9: Top-ranked scale decomposition for the first sensor of the Dike data.

dent Component Analysis [18] aims at separating a set of signals from a set of mixed signals but, in its standard formulation, requires at least as many sensors as sources. Our method is able to operate on a single input sensor and a variable number of sources to be discovered. Megalooikonomou et al. [69] introduce a multi-scale vector quantized representation of time series which enables fast and robust retrieval. The considered scales are however predefined and our approach could be used as a preprocessing step to determine those to include in the dictionary. The Minimum Description Length principle has been applied to the problem of choosing the best representation for a given time series by Hu et al. [42]. The authors propose a method to choose the best representation (and its parameters) among APCA, PLA and DFT. While there are similarities with our method (we also use the MDL principle to select the best model parameters for a given component), the authors put the stress on discovering the intrinsic cardinality of the data, other than its constituent multi-scale components. MDL has also been adopted to detect changes in the distribution of a data stream by van Leeuwen et al. [94].

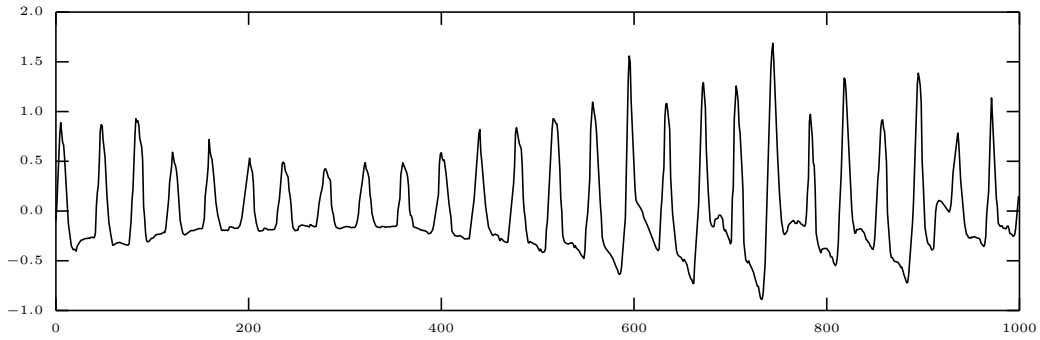


Figure 4.10: Detail of the first component.

In [44], the authors propose a method to select non-linear models from data, possibly generated by chaotic systems, having a good robustness to noise. Again, this work does not take into account the eventual multi-scale nature of the data.

4.6 Conclusions and Future Work

We introduced a novel methodology to discover the fundamental scale components in a time series in an unsupervised manner. The methodology is based on building candidate scale decompositions, defined over the scale-space image [103] of the original time series, with an MDL-based selection procedure aimed at choosing the optimal one.

A useful side product of the presented technique, due to the adoption of MDL, is that each discovered component is represented independently according to its inherent complexity and often results in a cheaper model (in terms of MDL score) in relation to the original raw time series. These cheaper per-component representations may better serve tasks like classification, regression or association analysis for time series produced by inherently multi-scale physical and artificial systems.

We have shown that our approach successfully identifies the relevant scale components in both artificial and real-world time series, giving meaningful insights about the data in the latter case. Future work will experiment with

diverse representation schemes and hybrid approaches (such as using combinations of segmentation, discretization and Fourier-based encodings). Moreover, another interesting research question is how to substitute the presently employed exhaustive search of the optimal decomposition with a computationally cheaper heuristic approach, which is necessary in the case of large time series data.

Chapter 5

Mining Variable-Length Motifs at Multiple Scales

5.1 Introduction

This chapter is concerned with the discovery of temporal patterns in large time series, produced from physical sensors. In all but the most trivial applications, such sensor data will reflect the complexity of the physical system under investigation, and will show a combination of multiple effects. Some of these effects will be of interest, and central to the sensoring system, but others, such as noise and environmental effects, will merely be a disturbance and a hindrance to the identification of the phenomena of interest. The complex physical systems we aim to investigate here often have two important characteristics: a) multiple phenomena are at play in the sensor signal, and they typically occur at different time scales, b) each phenomenon will involve recurring events that will show up in the signal as repeating segments of data, often deformed and warped. In this chapter, we introduce a method that elegantly combines these two characteristics in order to discover recurring events at multiple time scales.

As a motivating example, we consider again the bridge data from the InfraWatch project. In fact, such data fits our topic well as it is subject to a

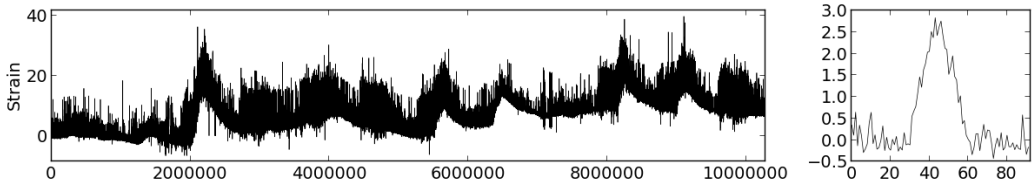


Figure 5.1: The plot on the left shows twelve days of strain measurements, sampled at 10 Hz from the Hollandse Brug bridge. The data exhibits recurring events, often superimposed, at multiple time scales, such as individual vehicles, traffic jams and daily fluctuations due to temperature changes. On the right it is shown an example of motif due to a passing vehicle.

number of effects that both show recurring events (traffic, daily temperature cycles), as well as largely varying time scales. The vertical displacement of the bridge, measured through *strain gauges*, is of course dependent on individual vehicles passing the bridge, over a period of several seconds (several tens of measurements). On a medium scale, the strain signal will show traffic jams, lasting up to an hour, that appear as clearly delineated intervals where the strain is increased due to the higher number of vehicles on the bridge. Finally, on a large scale, the strain is highly sensitive to the temperature of the bridge, such that the signal is dominated by a slow movement of the baseline, most notably with a day/night rhythm. Figure 5.1 shows 12 days of data collected at this bridge (some 10 million readings). Note that these different effects appear in a mixed fashion, and events at different time scales will often overlap. For example, traffic jams and individual vehicles will simply appear superimposed on the continually changing baseline of temperature effects on the strain. Additionally, vehicle peaks (shown as a detail on the right) will appear in the signal, even during traffic jams, as these often only affect one direction of traffic.

The recognition of repeating phenomena in time series is an important task in many applications, as it enables further processing of the data at a more conceptual level. For example, in SHM, it permits determining traffic load statistics or various load-induced vibration patterns because it is vital to know when exactly certain events occur (such as heavy trucks). We assume

that the recurring events will appear in a relatively small set of classes (e.g. trucks, cars), which we will refer to as *motifs*. The (scale-aware) motif discovery method presented here will then determine what the relevant motifs are, and when the different instances of each motif occur. In the specification of motifs, we intend to allow for a certain degree of flexibility in terms of duration and magnitude of the event. For example, a truck will be recognized as such, despite minor variations in speed and weight of the truck. Note that our definition of 'motif' is somewhat different from the use in other papers dealing with a similar problem [74], where a more strict matching based on Euclidean distance of a segment of fixed duration is employed. Although this approach works well in many scenarios, more flexibility is needed in the applications we consider.

In the motif discovery task in complex data, an important challenge we deal with is the possibility of superimposed events. Instances of motifs in one scale will overlap those in other scales, and the recognition of similar instances will be disturbed, if the possibility of multi-scale interference is not taken into account.

In this work, we propose an approach based on scale-space images [103] and the *minimum description length* (MDL) principle to address this problem [34]. The reason for choosing an MDL-based approach is that it allows us to find sets of motifs that represent a good trade-off between representation power and model simplicity. This guarantees that the reported motifs are actual recurring phenomena, rather than accidental coincidences, and that the motifs found are not too similar to each other. The novelty of our code, compared to an earlier approach [78] to the same problem, is that it explicitly supports the discovery of, potentially overlapping, multi-scale motifs.

The main contribution of this work is an algorithm for effectively finding multi-scale motifs that score well with respect to the MDL principle. Our algorithm combines several key ideas to achieve this:

- it uses *scale-space images* to characterize the contribution of the motifs at different temporal scales;
- it uses the *zero-crossings of derivatives* of the time series at different

scales to identify repeating linear segments in the time series;

- it uses a *a symbolic representation* in combination with suffix trees to identify promising motifs consisting of these linear segments;
- it uses a greedy algorithm to select characteristic motifs that score well with respect to an MDL score.

We evaluate our method on a number of sensor-based time series from various applications. Results show that our approach can effectively discover a small set of characteristic motifs in the data, often directly related to particular events in the corresponding application domain.

The structure of this chapter is as follows. Section 5.2 will introduce the notation and present necessary background information, including MDL, and the problem statement. In Section 5.3, we will motivate and define our method. Section 5.4 will evaluate and discuss experimental results. Section 5.5 presents related work. Finally, in Section 5.6, we draw conclusions and present ideas for future work.

5.2 Background and Problem Setting

In this section, we introduce the notation, provide necessary background information and formally define the problem.

5.2.1 Notation and Preliminaries

We deal with finite sequences of numerical measurements (samples), collected by observing some property of a system with a sensor and represented as time series as defined in 3.1. Moreover, and without loss of generality, we assume that the values are collected at a constant rate and none of them are missing and that the data has been z-normalized.

As motivated in the introduction, our goal is to find characteristic motifs in the input time series at multiple temporal scales. There are two equiva-

lent ways of looking at motifs. The first is that a motif is a structure that approximately repeats itself in a large number of places in the time series. The second is that a motif is a set of subsequences in the data, each pair of which is similar to each other [74]. We will refer to a structure that is approximately repeated in the data as a *motif*; subsequences of the data in which this motif occurs are referred to as *motif instances*.

An important feature of the motifs that we are looking for is that their instances can be warped or deformed to deal with potential slight variations in the duration and intensity of the events. This motivates our choice to represent motifs using linear segments as follows.

Definition (Motif). A **motif m** is a sequence of linear segments

$$[(a_1, b_1), (a_2, b_2), \ldots, (a_k, b_k)]$$

where a_i indicates the length of a segment (the duration) and b_i indicates the difference in value between the begin and end points of the segment.

In principle, higher order polynomials or other more complex functional representations may also be used to represent the segments, but we found that linear segments are simpler, have the advantage of avoiding overfitting, and are accurate enough in most cases.

We will be looking for instances of these motifs in the data.

Definition (Motif Instances). Given a set of motifs M, let I be a function that maps a motif $\mathbf{m} \in M$ and a segment t of this motif to a set of subsequences of $\mathbf{x}$:

$$I(\mathbf{m}, t) = \{\mathbf{x}[a_{1t} : b_{1t}], \ldots, \mathbf{x}[a_{kt} : b_{kt}]\}$$

for some $a_{it}, b_{it} \in \{1, \ldots, n\}$ such that $a_{it} = b_{i(t-1)} + 1$ for $t > 1$ (i.e., the end of a segment determines the start of the next segment). Then $I(\mathbf{m}, t)$ determines the set of **instances** in $\mathbf{x}$ of segment t of motif $\mathbf{m}$.

Some choices for I are better than others; ideally instances closely resemble their associated motifs. The MDL score introduced in the next section will be used to evaluate the quality of a set of motifs M and of a function I.

Note that subsequences for the same motif and motif segment can have different lengths. This is necessary to deal with time warping.

From a high-level perspective, the problem that we are interested in is to identify a set of motifs that characterizes the data well. Taking into account the multi-scale nature of the data, it is desirable that instances of different motifs can overlap. In this way, one motif can reflect a regularity at a coarse scale, and another can reflect a regularity at a finer scale superimposed on top of the coarse structure. The next section defines more precisely how we evaluate a set of motifs and its instances to reflect these requirements.

5.2.2 Minimum Description Length

Our main idea is to approach the problem of selecting motifs as a *model selection* problem. This allows us to employ the Minimum Description Length [34] principle, introduced in Section 3.4, to rank motifs.

In our setting, a model consists of a set of motifs M. Following the two-part MDL principle, the best set of motifs to describe the time series $\mathbf{x}$ is the one that minimizes the sum $L(M) + L(\mathbf{x} \mid M)$, where

- $L(M)$ is the length, in bits, of the description of the motifs, corresponding to a model;
- $L(\mathbf{x} \mid M)$ is the length, in bits, of the description of the time series when encoded with the help of the motifs M, that is the residual information not represented by M.

In order to apply the MDL principle in practice, the input data has to be discretized. We do this as discussed in Section 3.4.1. Moreover, we need to define an encoding scheme for a given set of motifs M and, consequently, how to compute both $L(M)$ and $L(\mathbf{x} \mid M)$. These aspects are addressed in the following sections.

Encoding of the model

We will first discuss the encoding of the model, i.e. a set of motifs M. Each motif essentially consists of a sequence of linear segments, each described by two integers. The length of a segment cannot be longer than the total length of the time series; hence, we use $\log_2 n$ bits to encode it. The difference in value between the begin and end point is limited by the quantization used; in our setting 8 bits are sufficient. Finally, with $\log_2 n$ bits we can encode the number of segments in a motif. Summing up we have

$$L(M) = \sum_{i=1}^{m} (\log_2 n + k_i(\log_2 n + 8))$$

where m is the number of motifs and k_i is the number of segments in motif i. We assume that these motifs are ordered in the encoding. We use this order to distinguish the scales at which the motifs are present.

Encoding the data

We will now describe how we compute $L(\mathbf{x} \mid M)$, that is the description length of the time series when encoded with the help of a set of motifs M. In the definition of the code, we will also use the instances I associated to each motif in M. Our assumption is that a good selection of motifs M and associated instances I will help to encode the data more concisely.

We will first define the entropy of a time series as it is a key concept we will need in the following paragraphs.

Definition (Time Series Entropy). The **entropy** of a time series $\mathbf{x}$, discretized according to a set of values D, is defined as below

$$H(\mathbf{x}) = -\sum_{v \in D} P(\mathbf{x}[i] = v) \log_2 P(\mathbf{x}[i] = v)$$

where $P \log_2 P = 0$ in the case of $P = 0$ and $P(\mathbf{x}[i] = v)$ indicates the fraction of points in the time series which has value v.

Given the definition of entropy, we can define the description length of a time series as follows, assuming we have not identified any motifs.

Definition (Time Series Description Length). Given a time series $\mathbf{x}$ of length n, its **description length** (in bits) is given by

$$L(\mathbf{x}) = nH(\mathbf{x})\,.$$

Our main idea is now that a good choice of motifs M and associated instances $I(\mathbf{m}, t)$ should lead to a code for the time series with a description length shorter than $L(\mathbf{x})$. To this aim, we introduce a code for $\mathbf{x}$ given a choice of M and $I(\mathbf{m}, t)$. It will be the task of the search algorithm to determine the best configuration.

Concretely, for each chosen motif $\mathbf{m}$ and corresponding motif instances $I(\mathbf{m}, t)$, we first encode the time stamps and the (vertical) values at which the instances of the first segment of $\mathbf{m}$ start. For one motif with ℓ instances, this requires $\log_2 n + \ell(\log_2 n + 8)$ bits, where $\log_2 n$ bit are needed to encode the number of instances, and $\log_2 n$ and 8 are the bits needed to code the starting time stamp and vertical value, respectively.

There are instances for each segment in a motif. While encoding these, we need to allow for a certain amount of time warping, and hence the segment in each instance may deviate both in length and in amplitude from the segment in the motif. Both vertical and horizontal differences from the segment in the motif can be represented by sequences of integers: the deviations of segment lengths can be represented in one sequence

$$[a_{ijk} \mid 1 \leq i \leq m, 1 \leq j \leq \ell_i, 1 \leq k \leq k_i]$$

where m is the number of motifs, ℓ_i the number of instances of motif i and k_i the number of segments in the motif; similarly, the differences in value can be listed. In order to favor only small numbers of values, we compute the description length of these sequences employing an entropy-based encoding as in Definition 5.

This code for motifs and instances leads to an approximation of the data, as follows. For each position in the time series, we determine the last motif in the ordered set of motifs which has an instance at this position. Whether a position is covered by an instance is determined by taking into account the

starting positions of the first segment of the motif, the lengths of the other segments in the motifs, and the deviations from these lengths as encoded in the code of deviations. The reason for using the order of motifs is that we explicitly allow motifs to overlap. This allows us to deal with the multi-scale aspects of the data.

The approximated value of a position in the time series covered by a motif is determined by linear interpolation between the two end points of the motif segment in which the position is included. These end points are determined similarly from the encodings of locations, motifs and deviations.

Our remaining code for the data now consists of two parts. First, for each position in the data covered by a motif, the error is encoded with respect to the approximation. An entropy encoding is used for these errors. Second, for the remaining time stamps, which are not covered by a motif, an entropy encoding is used as well to code the original value for that position.

Note that in this code we have a constant number of dictionaries (for duration, difference in value, errors, and remaining original time points). Hence, we do not need to calculate the size of these dictionaries explicitly.

The final description length $L(\mathbf{x} \mid M)$ is given by the sum of the lengths (in bits) of the code components described above.

5.2.3 Problem Statement

We have now introduced the necessary definitions and background material to state our problem.

> Given a time series $\mathbf{x}$, we want to find a set of motifs M and associated instances, such that the sum $L(M) + L(\mathbf{x} \mid M)$ is minimized.

Clearly, this problem is hard to solve exactly. Hence, in the next section we define a step-wise heuristic algorithm that works well in practice.

5.3 Motif Selection Algorithm

The proposed heuristic motif discovery algorithm consists of several steps, which will be shown to perform well in the next section. The first steps will identify a set of promising candidate motifs; the last steps select a characteristic subset of the motifs based on the MDL scoring function discussed earlier. Figure 5.2 shows a high level overview of our method and the steps involved.

5.3.1 Finding Candidates Motifs

In this section, we describe our candidate motif generation procedure. Several key ideas underly this procedure.

- It uses the *scale-space image* to characterize the contribution of the motifs at different temporal scales;
- It effectively identifies promising segments at multiple scales by discretizing the time series using *the derivatives of the signal in scale-space* in combination with *k-means clustering*;
- In the discretized representation, it merges recurring sequences of adjacent segments by employing a *suffix tree* based approach.

The subsequent sections discuss this in more detail.

Scale-Space Image

We rely on the concept of scale-space image introduced in Section 3.3. As noted, in practice the scale-space image is quantized along the scale dimension by computing the Gaussian convolution only for a limited number of scale parameters. The number of scale parameters considered, and thus the resolution of the quantization, depends on the final application and on the distribution of the motifs across the scale dimension. If, for instance, the motifs appear at considerably different scales, a coarser quantization would

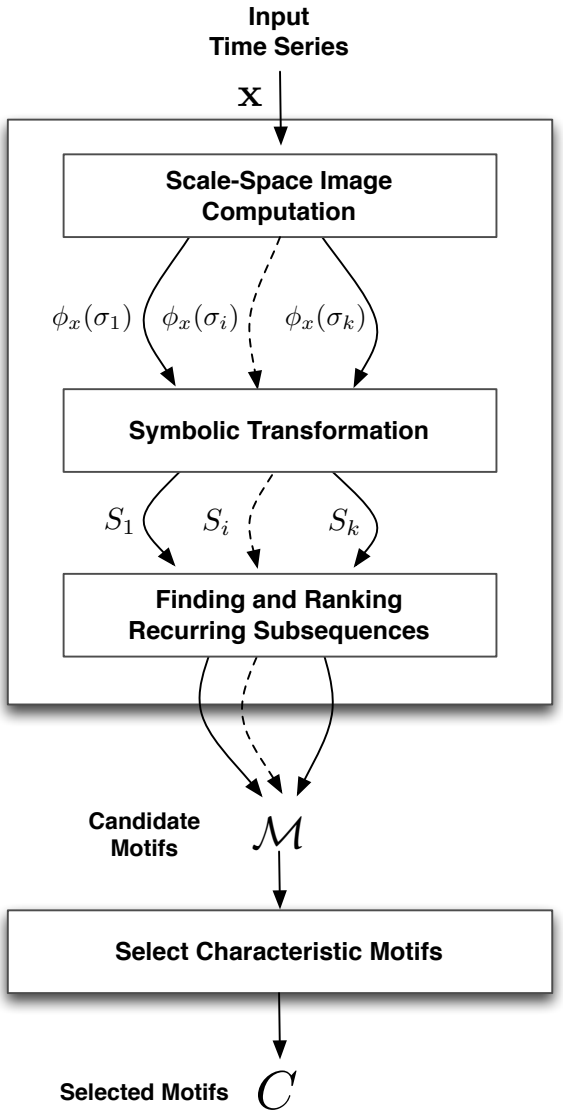


Figure 5.2: Overview of the proposed motif discovery and selection algorithm.

suffice to isolate them across the scale dimension. On the other hand, if different motifs appear at similar scales, a finer quantization is needed to effectively separate their corresponding contributions to the signal.

In order to support both scenarios, we define two sets of scale parameters $S_{coarse} = \{2^i \mid 0 \leq i \leq \sigma_{max} \ \wedge i \in \mathbb{N}\}$ and $S_{fine} = \{\sqrt{2^i} \mid 0 \leq i \leq 2\sigma_{max} \ \wedge i \in$

$\mathbb{N}\}$ which well adapt to the practical cases we consider.

We deal with multi-scale aspect of the data by identifying motifs in each of the scales in the scale image.

Finding candidate segments

Before identifying candidate motifs, we first identify candidate linear segments. A useful tool to quickly identify promising boundaries for linear segments in the time series are the *zero-crossings of derivatives*.

Given a time series $\mathbf{x}$ and one of the components of its scale-space image $\Phi_{\mathbf{x}}(\sigma)$, let

$$z(j) = \{t_1, \ldots, t_m\}, \text{such that } \frac{d^j \Phi_{\mathbf{x}}(\sigma)}{dt}(t_i) = 0,$$

$$Z = z(1) \cup \cdots \cup z(d_{max})$$

be the sorted locations in $\Phi_{\mathbf{x}}(\sigma)$ of the zero-crossing of its derivatives until order d_{max}. Note that d_{max} will typically be low, e.g. just 1 or 2.

These zero-crossings are informative as they indicate points in the time series at which the direction of the signal changes; these positions are good candidates for a change of the linear coefficients as well. Thus, each segment bounded by two consecutive zero-crossings could be an instance of a segment in a motif. We use $k-$means clustering to identify a small set of prototype segments, as follows. Each segment between zero-crossings can be thought of as a data point in a feature space, where the features are the duration and difference in value between the zero-crossings of the derivatives. More precisely, we consider the data points $F_{\Phi_{\mathbf{x}}(\sigma)} = \{\mathbf{f}_i = (h_i, v_i)\}$ where

$$h_i = t_{i+1} - t_i \, , \, 1 \leq i < n$$

is the time between each pair of consecutive zero-crossings and

$$v_i = \Phi_{\mathbf{x}}(\sigma)[t_{i+1}] - \Phi_{\mathbf{x}}(\sigma)[t_i] \, , \, 1 \leq i < n$$

is their vertical distance. Figure 5.3 illustrates this concept.

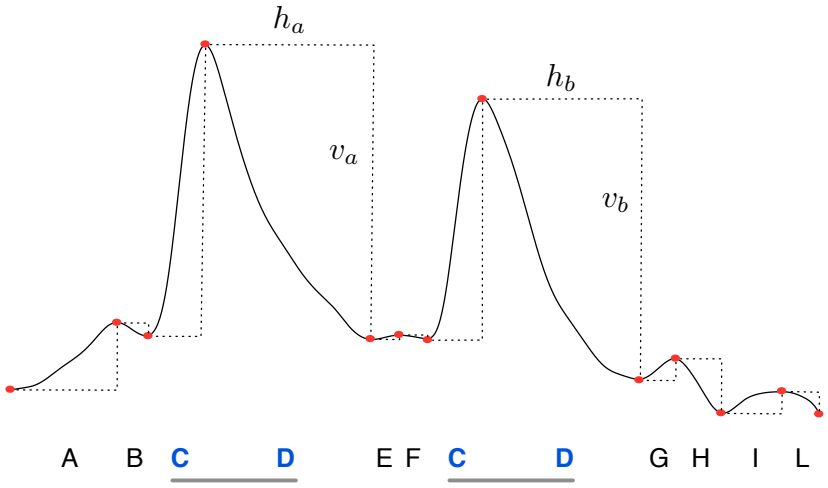


Figure 5.3: Example of the feature space based on the zero-crossings of the derivatives (only order one in this figure) and the clustered candidate segments identified by letters.

These data points are clustered using the $k-$means clustering algorithm, where k is a parameter that determines the number of candidate segments. Preliminary experiments show that setting the parameter k in practice is not a critical problem.

The centers of the identified clusters are the candidate reference segments, which will be combined into motifs in the next step. Note that the clustering algorithm ensures that candidate segments will be not too dissimilar from each other. This procedure is repeated for each scale in the scale-space independently.

Finding candidate motifs

The key idea in identifying motifs is to represent time series symbolically (see Figure 5.3). Each symbol in this representation corresponds to the candidate segment identified by the $k-$means algorithm for that segment.

After transforming each scale-space image component into the symbolic representation defined above, we identify motifs by looking for repeating subse-

quences in the obtained string as similarly done by previous approaches [50, 60], although using different kinds of representations such as SAX [61].

Algorithm 1 Find candidate motifs

Require: a time series $\mathbf{x}$, a set of scales parameters $S = \{\sigma_1, \ldots, \sigma_k\}$, the maximum order for the derivatives roots d_{max}, the cardinality A of the symbolic representation, the number of motifs considered per scale r

Ensure: a set of candidate motifs $\mathcal{M} = \{M_{s,r}\}$ indexed by scale parameter s and rank r.

$\mathcal{M} = \{\}$

$\Phi_{\mathbf{x}}(\sigma_1), \ldots, \Phi_{\mathbf{x}}(\sigma_k) = \texttt{ScaleSpaceImage}(\mathbf{x}, S)$

for $i = 1 \ldots k$ **do**

$Z_i = \texttt{ComputeZeroCrossings}(\Phi_{\mathbf{x}}(\sigma_i), d_{max})$

$S_i = \texttt{SymbolicQuantization}(\Phi_{\mathbf{x}}(\sigma_i), Z_i, A)$

$\Sigma_i = \texttt{FindRecurringSubstrings}(S_i)$

$M_{\sigma_i,r_1}, \ldots, M_{\sigma_i,r_m} = \texttt{RankMotifsByCoverage}(\Sigma_i, r)$

$\mathcal{M} = \mathcal{M} \cup \{M_{\sigma_i,r_1}, \ldots, M_{\sigma_i,r_m}\}$

end for

Our candidate motifs generation procedure is summarized in pseudo-code in Algorithm 1.

$\texttt{ScaleSpaceImage}(\mathbf{x}, S)$ returns the scale-space image of $\mathbf{x}$ defined over the scale parameters S.

$\texttt{ComputeZeroCrossings}(\Phi_{\mathbf{x}}(\sigma_i), d_{max})$ calculates the zero-crossings of the derivatives for each scale.

$\texttt{SymbolicQuantization}(\Phi_{\mathbf{x}}(\sigma_i), Z, A)$ transforms each time series $\Phi_{\mathbf{x}}(\sigma_i)$ into a symbolic string given the zero-crossings Z and cardinality A.

$\texttt{FindRecurringSubstrings}(S_i)$ returns the set of all maximal substrings of length at least 2 that appear at least twice in the data (maximal in the sense that no longer substring occurs twice). In general, we could parameterize this; however, in our experiments we found these parameters to work in all cases. Furthermore, an important advantage of this setup is that we can calculate this set of substrings in linear time by using suffix trees.

`RankMotifsByCoverage`(Σ_i, r) selects the best scoring r motifs from this set of substrings. The evaluation is as follows: the occurrences of each string in the time series are determined; these occurrences are mapped back to the original time series; the total length of the original time series covered by these occurrences is determined. The main motivation is that we can expect the best coding motifs to be those that cover large parts of the time series. The final selection from the resulting set of candidate motifs is done in the next step.

5.3.2 Selecting Characteristic motifs

The naive way to select the best set of motifs would be to enumerate all potential subsets and choose the one that minimizes the sum $L(M) + L(\mathbf{x} \mid M)$. However, the space of motif sets grows exponentially with the number of candidate motifs and this makes an exhaustive evaluation computationally infeasible for large time series. Because of this, we propose a heuristic selection strategy that overcomes these computational limitations. Our motif selection heuristic is shown in pseudo-code in Algorithm 2.

Algorithm 2 Select characteristic motifs

Require: a time series $\mathbf{x}$, a set of candidate motifs $\mathcal{M} = \{M_{s,r}\}$ indexed by scale parameter s and rank r.

Ensure: a set of selected motifs $C \subseteq \mathcal{M}$.

$C = \{\}$

for $i = k \ldots 1$ **do**

$$j = \arg \min_{j \in \{1,\ldots,m\}} L(C \cup \{M_{\sigma_i,j}\}) + L(\mathbf{x} \mid C \cup \{M_{\sigma_i,j}\})$$

$C = C \cup \{M_{\sigma_i,j}\}$

end for

Essentially this algorithm traverses the candidate motifs starting at the coarsest scale and, for each scale, it adds the motif that improves the MDL score the most.

5.3.3 Computational Complexity

The construction of the scale-space image requires to compute $|S|$ convolutions. This can be done efficiently using the Fast Fourier Transform in $O(|S|\, n \log_2 n)$ time. The computation of the zero-crossing of the derivatizes can be done with a linear scan and thus has $O(n)$ complexity. The complexity of the symbolic transformation, carried out by k-means in $O(Ik\,|Z|)$ time depends on the number of zero-crossings features to cluster which, given a property of the scale-space image [103], can only decrease as the scale is increased; here I is the number of iterations of the $k-$means algorithm. Preliminary experiments even show that the decrease in $|Z_i|$ is exponential. Locating recurring substrings in the symbolic representation can be done in linear time employing a suffix tree; the number of such strings ($|\mathcal{M}|$) is $O(n)$ in the worst case and much smaller in practice. We calculate the instances of the corresponding motifs in $O(n)$ time for each motif identified. Sorting the resulting motifs takes $O(|\mathcal{M}| \log |\mathcal{M}|)$ time. During the final traversal of this set, we need to calculate the MDL score for each intermediate model. This calculation takes $O(|C|n)$ time; note that the size of the dictionaries can be considered constant. Overall, this gives our method a complexity of $O(n \log_2 n + |\mathcal{M}|(\log |\mathcal{M}| + |C|n)$ time.

5.4 Experimental Evaluation

In this section, we evaluate our method experimentally, on two real-life sensor datasets, one describing physical exercise, and one collected from the sensor network of the highway bridge mentioned in the introduction. Moreover, we compare our method with another published approach on a common dataset.

5.4.1 Snowboard Data

The first experiment relates to physiologic data collected during a day of snowboarding in the Austrian Alps. The data was collected by a Zephyr Bio-

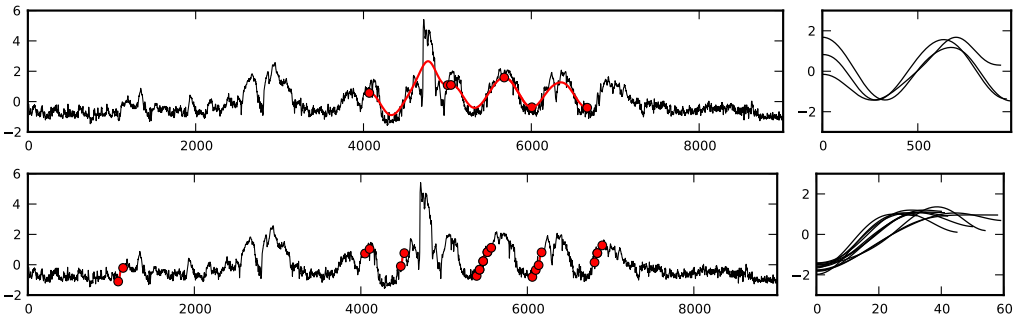


Figure 5.4: Selected motifs in the Snowboard data. Left side: motif occurrences in the series. Right side: motifs at the respective scale-space component after z-normalization.

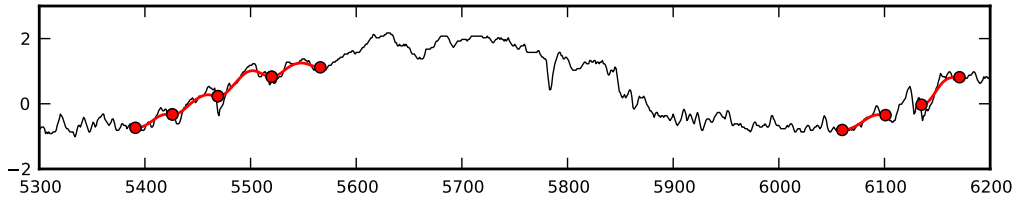


Figure 5.5: Detail of the short-term motifs depticted in Figure 5.4 (bottom).

Harness[1] 3 breast strap, which monitors several key physiological parameters and logs them at a sampling rate of 1 Hz. Alpine sports are an interesting domain for our method, as it naturally contains the cyclic phenomenon of ascending by ski lift and descending 'on foot'. This produces a recurring pattern of intense exercise while descending and clear signs of recuperation while being transported up. Especially when the same lift and slope are repeatedly taken, this will lead to motifs in the measured time series. Additionally, on a smaller scale, the natural tendency of the human body is to introduce shorter cycles of activity and rest, especially when dealing with intense activity and high altitude.

The data considered here describes heart rate measurements taken during 2.5 hours of mixed activity, starting at 11:00 AM, with some 40 minutes actually spent on the slopes. We employed S_{fine} as scale parameters, set $d_{max} = 1$ and the cardinality of the symbolic representation to 10. Figure 5.4 shows two

[1] http://www.zephyranywhere.com/products/bioharness-3

key selected motifs, which correspond to the phenomena described above. The top motif represents some 16 minutes, corresponding to recuperation (decreasing heart rate while on the lift), exercise and recuperation again. A full cycle of ascent and descent takes about 10 minutes, which corresponds with the manual annotations. This pattern occurs three times in this dataset, at the scale component $\Phi_x(\sqrt{2^{14}})$, as indicated by the red segments in the diagram. Note that two instances actually overlap, as the motif describes more than a single cycle. These two instances actually relate to two descents of a single slope. The second motif, at the scale component $\Phi_x(\sqrt{2^7})$, has 10 instances of increasing and then decreasing heart rate, presumably related to short exercise intervals of around 50 sec. A detail of this motif is shown in the bottom diagram, showing just 20 minutes at 12:25.

The overall number of scale components considered for this data is 22 for a total of 13 selected motifs. However, motifs selected at scales greater than 2^{16} did not show motifs relevant to this particular application domain.

5.4.2 Highway Bridge Data

We subsequently evaluate our approach on the time series data previously shown in Figure 5.1. The series has been collected in the context of a Structural Health Monitoring project,and consists of 12 days of strain measurements (for a total of $10,280,939$ data points) from one span of the monitored highway bridge. As the bridge is affected by several phenomena operating at multiple time scales, the strain measurements contain various classes of recurring motifs reflecting this fact and represents an ideal dataset to test our method. We employed S_{coarse} as scale parameters and set $d_{max} = 1$ and the cardinality of the symbolic representation to 10. Figure 5.6 shows two of the most interesting selected motifs, respectively at scale components $\Phi_x(2^3)$ and $\Phi_x(2^{15})$. The first motif identifies the most recurring events in the data, i.e. passing vehicles. In the graph, a red pixel is drawn for each instance, for a total of $58,646$ occurrences, which cover almost 22% of the data. On the right, we plot all the motif instances (after normalization) superimposed, as represented in the scale component $\Phi_x(2^3)$. The selected motif represents a

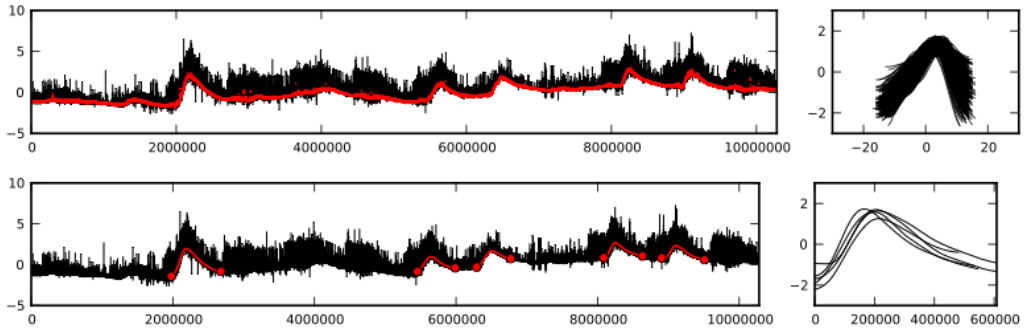


Figure 5.6: Selected motifs in the highway bridge data. Left side: motif occurrences in the series. Right side: motifs at the respective scale-space component after z-normalization.

high variability of instances, in both duration and amplitude, that can be directly related to the speed and weight of the vehicles. This information can thus be used by bridge managers to evaluate the load patterns of the infrastructure and potentially aid the decision making when planning maintenance activities. The second motif represents a much longer pattern occurring on a daily basis due to changes in temperature that, in turn, affect the response of the bridge to external forces. A total of 5 motif instances of this kind occur, covering around 24% of the data. Note how occurrences of the first motif are superimposed over the instances of this one.

The overall number of scale components considered is 19, although the motifs selected at scales greater than 2^{17} are not of any interest in relation to the application domain.

5.4.3 Comparison with Related Work

To the best of our knowledge, there are no published methods dealing with the discovery of characteristic sets of multi-scale and overlapping motifs in time series data. As we cannot compare our method with others in a multi-scale setting, we chose to also evaluate our algorithm on a time series presented in [78], in which no multi-scale events are present. A comparison on such data is of interest as our method should be able to identify the non-

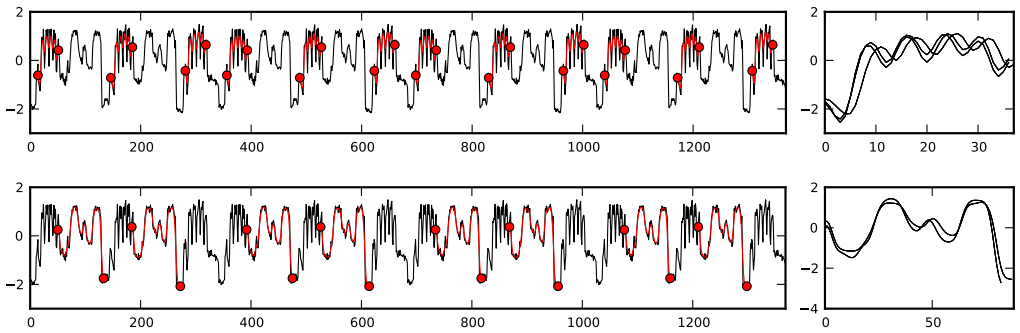


Figure 5.7: Selected motifs in the *bird calls* data from [78]. Left side: motif occurrences in the series. Right side: motifs at the respective scale-space component after z-normalization.

overlapping motifs present in this data as well.

The considered time series was produced by extracting the first MFCC coefficients from an audio file featuring two repeated kinds of bird calls, resulting in two motifs present in the data. The time series has a total of 1367 measurements. As the motifs in the data are rather similar in length, we do not need to consider the whole scale-space image. Instead, we set the scale parameters to $S = \{1, \sqrt{2}, 2, \sqrt{8}\}$. The result shown here was obtained by setting the cardinality of the symbolic representation to 6. However, in order to assess the sensitivity of the method in relation to the size of the alphabet, we tried cardinalities ranging from 5 to 15 obtaining qualitatively similar results. Figure 5.7 reports the motifs selected by our method. These motifs are similar to those obtained by the clustering method proposed in [78] for non-overlapping motifs. Although in this case we manually specified the scale parameters, we note that the algorithm in [78] also requires to provide an educated guess of parameters, i.e. of the approximate lengths of the motifs to look for.

5.5 Related Work

The problem of discovering recurring temporal patterns in time series data is an important one and has received considerable attention by the community from different perspectives.

Subsequence Clustering. Early work considers the related problem of clustering the (overlapping) subsequences in the time series extracted through a sliding window. Subsequence clustering is an obvious and intuitive choice for finding characteristic subsequences in time series. However, this approach requires the a priori specification of the lengths of the subsequences to consider and is not generally tailored to support multi-scale data. Moreover, in a paper by Keogh et al. [49], it was shown that, despite the intuitive match, subsequence clustering is prone to a number of undesirable behaviors that makes the end result meaningless and independent of the data at hand. A number of papers [13, 22, 97] have further investigated the observed phenomena, providing solutions to overcome it. Yet, since the publication of [49], the subsequence clustering idea has seen a serious decline in popularity. In [49], the authors proposed a solution based on motif discovery.

Motif discovery and clustering. Motif discovery has received a fair amount of attention, in particular after subsequence clustering was shown to be unreliable. In [74], a motif is defined rather strictly as the pair of most similar subsequences in a time series according to the Euclidean distance, and the authors propose an efficient and exact method to find such pairs. Saria et al. [81], on the other hand, propose a more flexible definition of motif, based on a shape template that can be affected by non-linear transformations such as temporal warping and additive noise. They introduce an unsupervised algorithm to discover the set of canonical shape templates in the data. Although the method is able to discover motifs of different lengths, it does not deal with multi-scale data where multiple motifs at different time scales could appear superimposed.

To the best of our knowledge, the most similar work to ours is [78]. The authors propose a method to mine a set of clusters of motifs from a given

time series. The clusters are formed according to an agglomerative procedure. First, a single cluster is created containing the pair of most similar subsequences in the data (this is done with repeated runs of the exact motif discovery algorithm introduced in [74]). After that, the set of clusters is iteratively refined by taking one of the following actions: create a new cluster, add to a cluster, merge two clusters. The algorithm looks for the best operator to apply such that the MDL score for the clusters set is lowered, or it stops otherwise. This method does not however consider superimposed motifs like those found in the multi-scale data we consider in this chapter.

Multi-scale Time Series Data. Although several papers address the problem of discovering recurring patterns in time series, few of them consider data where combinations of effects at multiple temporal scales affect the patterns or motifs. In [76], Papadimitriou et al. propose a method to discover the key trends in a time series at multiple time scales (window lengths) by introducing an incremental version of Singular Value Decomposition. Vespier et al. [96] propose an MDL-based method to recognize the most relevant scales of analysis in the data and, consequently, to separate the time series into distinct components. This method does not however characterize the individual motifs directly, but rather assesses the relevancy on the informative content present at each temporal scale.

5.6 Conclusions and Future Work

In this chapter, we introduced a method for the discovery of multi-scale recurring patterns (motifs) in time series data. Our work is motivated by an SHM project which deals with high-frequency measurements collected by a sensor network deployed on a highway bridge. In particular, we focused on a property that sensor data collected from complex systems typically exhibits: the presence of multiple phenomena at play in the sensor signal, often occurring at different time scales and potentially superimposed and mixed together. Because of the high degree of variability present in this kind of data, we have adopted a definition of motif based on structural complexity other than on

point-wise similarity (i.e. Euclidean distance) as in much previous work. In order to discover the most characteristic recurring motifs, we proposed an algorithm based on a combination of scale-space theory, string processing and the Minimum Description Length principle. We showed the effectiveness of our method on sensor data from several applications.

Future work includes evaluating our method on additional data exhibiting multi-scale behavior, as a few datasets of this kind are currently publicly available. Moreover, we are interested in further developing the symbolic representation we adopted, currently requiring the cardinality of the alphabet as a parameter; ideally, our method would become parameter free.

Chapter 6

Subsequences Clustering for Events Modeling

6.1 Introduction

In this chapter, we investigate how to build a model of traffic activity events, such as passing vehicles or traffic jams, from measurements data collected in the context of Infrawatch, the Structural Health Monitoring project discussed in Section 2.4.

It has been shown that the structural stress caused by heavy loads is one of the main causes of bridge deterioration. Because of this, we focus here on modeling traffic activity events in the strain measurements, such as passing vehicles or traffic jams. The produced model can then be employed for real-time event classification or detection of anomalous responses from the bridge.

A single moving vehicle is represented in the strain measurements as a bump-shaped peak (see Figure 6.1 (right)) with an intensity proportional to the vehicle's weight and a duration in the order of seconds. On the other hand, events like traffic jams represent significantly larger time spans and cause an overall increase in the average strain level, due to the presence of many slow-moving vehicles on the bridge. Because we are dealing with events of

varying nature, straightforward algorithms based on peak detection will not suffice.

In order to model all the different kinds of traffic events represented in the strain data, we investigate the effectiveness of time series subsequence clustering [41, 49, 30, 43], which essentially employs a sliding window technique to split the data stream in individual subsequences (observations), which can then be clustered. However, the naive implementation of subsequence clustering (SSC) using a sliding window and k-Means is controversial, as it is prone to producing undesirable and unpredictable results, as was previously demonstrated and analyzed in several publications, e.g. [49, 30, 43]. Indeed, within our strain data application, we notice some of the mentioned phenomena, although not all. We provide an analysis of how the different phenomena can be explained, and why some of them are not present in the data we consider. Finally, we introduce a novel *Snapping* distance measure which, employed in SSC based on k-Means, removes the artifacts and produces a correct clustering of the traffic events. We believe that the proposed distance measure can lead to a rehabilitation of SSC methods for finding characteristic subsequences in time series.

6.2 InfraWatch and the Strain Sensor Data

In this section, we describe what the strain data looks like and how the different types of traffic activities are represented in the strain measurements, in order to motivate the technical solutions employed in Section 6.3.

As mentioned, we focus on modeling traffic events, such as vehicles passing over the bridge or traffic jams, represented in the strain measurements of the InfraWatch data. The data is being sampled at 100 Hz which amounts to approximately $8.6 \cdot 10^6$ measurements per sensor per day. As the sensor network is highly redundant, and the different strain sensors are fairly correlated or similar in behavior, we selected one sensor that is reliable and low in measurement-noise (less than 1.0 $\mu m/m$). The strain gauge considered is placed at the bottom of one of the girders in the middle of a 50 meter span

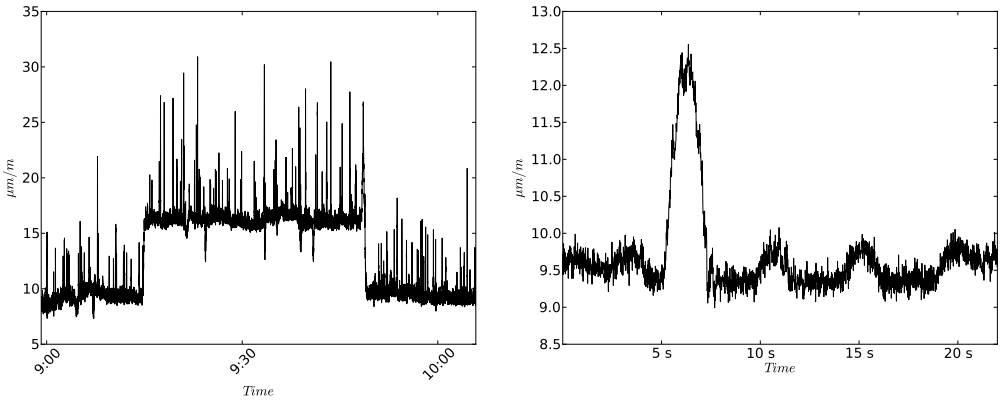


Figure 6.1: Detailed plots of strain, showing a traffic jam during rush hour (left) and individual vehicles (right).

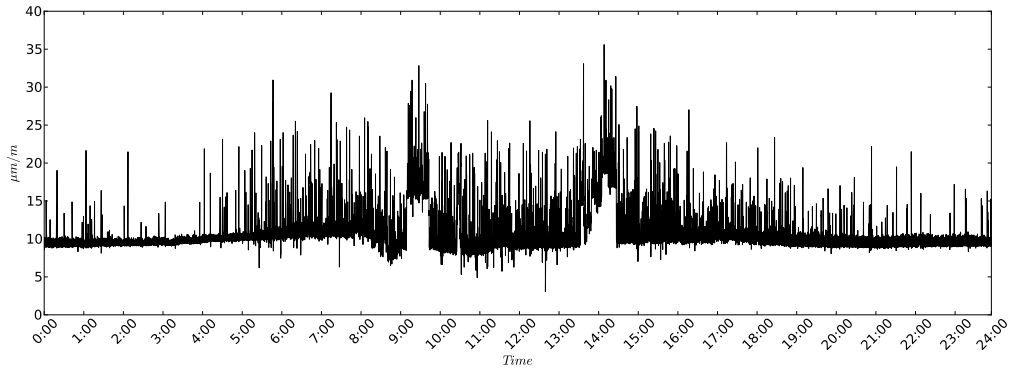


Figure 6.2: One full week day of strain measurements. All y-axis units in this chapter are in $\mu m/m$ (μ-strain).

near one end of the bridge. The strain data is thus related to this portion of the infrastructure. Every load situated on this span will have a positive effect, with loads in the middle of the span contributing more to the strain than loads near the supports of the span. Figure 6.2 shows an overall plot of the measurements for a single (week)day.

At the time scale of Figure 6.2, it is not possible to identify short-term changes in the strain level (except for notable peaks), such as individual vehicles passing over the span. However, long-term changes are clearly visible. For instance, there is a slightly curved trend of the strain baseline which slowly develops during a full day, which is due to changes in temperature,

slightly affecting both the concrete and gauge properties. The sudden rise of the average strain level between 9am and 10am is caused by a traffic jam over the bridge (as verified by manual inspection of the video signal). A traffic jam involves many slowly moving vehicles, which causes high vehicle densities. This in turn produces a heavy combined load on the span, and the strain measurements record this fact accordingly. Figure 6.1 (left) shows a detailed plot of the traffic jam event.

Short-term changes, on the other hand, can be identified when considering a narrower time window, in the order of seconds. A passing vehicle is represented in the data by a bump-shaped peak, reflecting the load displacement as the car moves along the bridge's span. Figure 6.1 (right) shows a time window of 22 seconds where the big peak represents a truck while the smaller ones are caused by lighter vehicles such as cars.

The examples above show how different traffic events, though all interesting from a monitoring point of view, occur with different durations and features in the strain data. Our aim is to characterize the different types of traffic the bridge is subjected to by analyzing short fragments of the strain signal, in the order of several seconds. The remainder of this chapter is dedicated to the clustering of such subsequences obtained by a sliding window.

6.3 Subsequence Clustering for Traffic Events Modeling

In this section, we introduce the rationale behind the subsequence clustering technique. We review the known pitfalls of SSC considering the features of the strain data and we show how its naive application produces results affected by artifacts. We finally propose a novel distance measure for SSC designed to remove the artifacts.

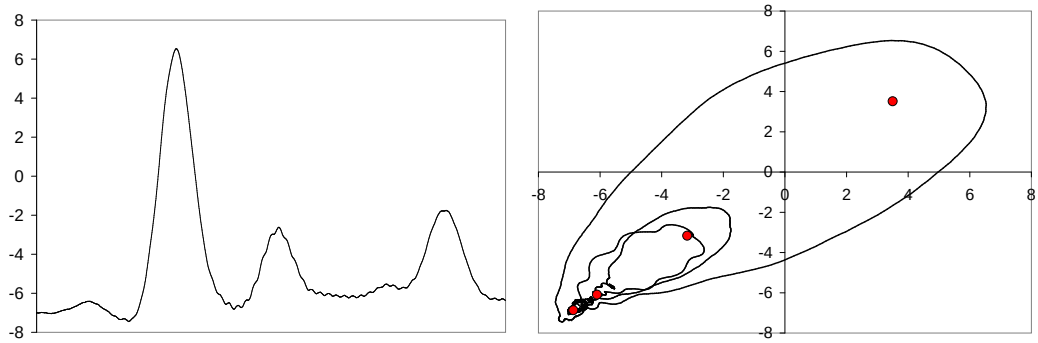


Figure 6.3: Two plots of the same data, showing the original data as a function of time (left), and a projection on two selected dimensions in w-space and the four prototypes generated by k-Means (red circles). Clearly, the sliding window technique creates a trajectory in w-space, where each loop corresponds to a bump in the original signal.

6.3.1 Subsequence Clustering

Subsequence clustering is a time series clustering techniques based on the concept of *subsequences set*:

Definition (Subsequences Set). The subsequences set $D(X, w) = \{S_{i,w} \mid 1 \leq i \leq m - w + 1\}$ is the set of all the subsequences extracted by sliding a window of length w over the time series X.

The subsequences set $D(X, w)$ contains all possible subsequences of length w of a time series X. The aim of *subsequence clustering* is discovering groups of similar subsequences in $D(X, w)$. The intuition is that, if there are repeated similar subsequences in X, they will be grouped in a cluster and eventually become associated to an actual event of the application domain.

6.3.2 Subsequence Clustering equals Event Detection?

Subsequence clustering is an obvious and intuitive choice for finding characteristic subsequences in time series. However, in a recent paper by Keogh et al. [49], it was shown that despite the intuitive match, SSC is prone to a number of undesirable behaviors that make it, in the view of the authors,

unsuitable for the task at hand. Since then, a number of papers (e.g. [43] and [30]) have further investigated the observed phenomena, and provided theoretical explanations for some of these, leading to a serious decline in popularity of the technique. In short, the problematic behavior was related to the lack of resemblance between the resulting cluster prototypes and any subsequence of the original data. Prototype shapes that were observed were collections of smooth functions, most notably sinusoids, even when the original data was extremely noisy and angular. More specifically, when the time series were constructed from several classes of shorter time series, the resulting prototypes did not represent individual classes, but rather were virtually identical copies of the same shape, but out of phase. Finally, it was observed that the outcome of the algorithm was not repeatable, with different random initializations leading to completely different results.

The unintuitive behavior of SSC can be understood by considering the nature of the subsequence set $D(X, w)$ that is the outcome of the initial sliding window step. Each member of $D(X, w)$ forms a point in a Euclidean w-dimensional space, which we will refer to as w-space, illustrated in Figure 6.3. As each subsequence is fairly similar to its successor, the associated points in w-space will be quite close, and the members of $D(X, w)$ form a trajectory in w-space. Figure 6.3 shows an example of a (smoothed) fragment of strain data, and its associated trajectory in w-space (only two dimensions shown). Individual prototypes correspond to points in w-space, and the task of SSC is to find k representative points in w-space to succinctly describe the set of subsequences, in other words, the trajectory. Figure 6.3 (right) also shows an example of a run of k-Means on this data. As the example demonstrates, the prototypes do not necessarily lie along the trajectory, as they often represent an (averaged) curved segment of it.

So how does SSC by k-Means fare on the strain data from the Hollandse Brug? Experiments reported in Section 6.4 will show that not all the problematic phenomena are present in clustering results on the strain data. In general, cluster prototypes do resemble individual subsequences, although some smoothing of the signal as a result of averaging does occur, which is only logical. The relatively good behaviour can be attributed to some cru-

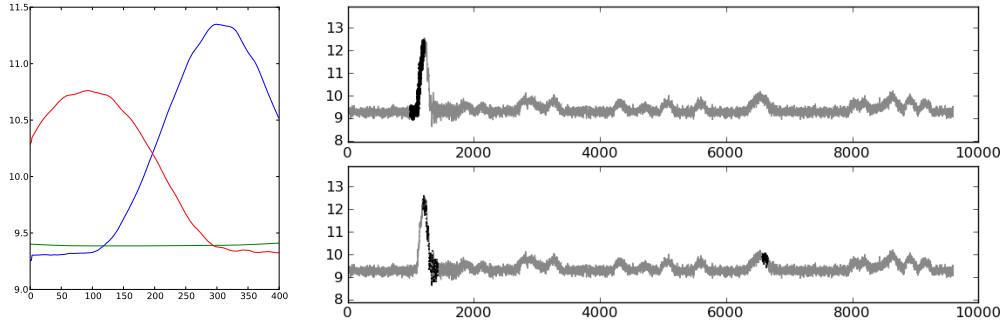


Figure 6.4: Multiple representation of events. The left plot shows the prototypes computed by the classic k-Means. The right plot shows, in black, the portion of the data assigned to the two bump-shaped prototypes.

cial differences between the nature of the data at hand, and that used in the experiments of for example [49, 43]. Whereas those datasets typically were constructed by concatenating rather short time series of similar width and amplitude, the strain data consists of one single long series, with peaks occurring at random positions. Furthermore, the strain data shows considerable differences in amplitude, for example when heavy vehicles or traffic jams are concerned. There remains however one phenomenon that makes the regular SSC technique unsuitable for traffic event modeling: the clustering tends to show multiple representations of what is intuitively one single event (see Figure 6.4 for an example). Indeed, each of the two bump-shaped prototypes resembles a considerable fraction of the subsequences, while at the same time having a large mutual Euclidean distance. In other words, our notion of traffic event does not coincide with the Euclidean distance, which assigns a large distance to essentially quite similar subsequences. In the next section, we introduce an alternative distance measure, which is designed to solve this problem of misalignment.

6.3.3 A Context-Aware Distance Measure for SSC

As showed in the previous section, applying SSC to the strain data employing the classic k-Means leads to undesirable multiple representations of the

same logical event. The problem is that comparing two subsequences with the Euclidean distance does not consider the similarity of their local contexts in the time series. Below we introduce a novel distance measure which finds the best match between the two compared subsequences in their local neighborhood.

Given a time series X and two subsequences $S_{p,w} \in X$ and S_{fixed} of length w, we consider not only the Euclidean distance between S_{fixed} and $S_{p,w}$, but also between S_{fixed} and the neighboring subsequences, to the left and to the right, of $S_{p,w}$. The minimum Euclidean distance encountered is taken as the final distance value between $S_{p,w}$ and S_{fixed}.

Formally, given a shift factor f and a number of shift steps s, we define the neighbor subsequences indexes of $S_{p,w}$ as:

$$NS = \left\{ \quad p + \frac{fw}{s} \cdot i \quad \middle| \quad -s \leq i \leq s \quad \right\}$$

The extent of data analyzed to the left and to the right of $S_{p,w}$ is determined by the shift factor while the number of subsequences considered in the interval is limited by the shift steps parameter. The *Snapping* distance is defined as:

$$Snapping(S_{p,w}, S_{fixed}) = \min\{Euclidean(S_{i,w}, S_{fixed}) \mid i \in NS\} \qquad (6.1)$$

We want to employ the *Snapping* distance in a SSC scheme based on k-Means. k-Means is a well known clustering/quantization method that, given a set of vectors $D = \{x_1, \ldots, x_n\}$, aims to find a partition $P = \{C_1, \ldots, C_k\}$ and a set of centroids $C = \{c_1, \ldots, c_k\}$ such that the sum of the squared distances between each x_i and its associated centroid c_j is minimized.

The classic k-Means heuristic implementation looks for a local minimum by iteratively refining an initial random partition. The algorithm involves four steps:

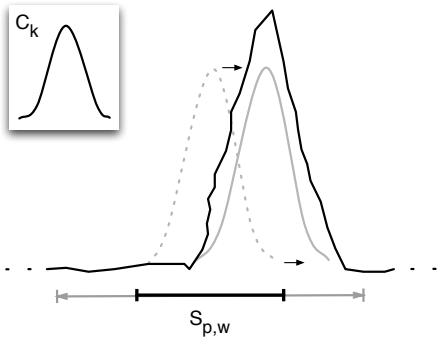


Figure 6.5: A subsequence $S_{p,w}$ is compared against the centroid C_k. The minimum Euclidean distance between C_k and the neighbor subsequences of $S_{p,w}$, including itself, is taken as a distance. Here, the best match is outlined in gray at the right of $S_{p,w}$.

1. (*initialization*) Randomly choose k initial cluster prototypes $c_1, \ldots, c_k$ in D.

2. (*assignment*) Assign every vector $x_i \in D$ to its nearest prototype c_j according to a distance measure. The classic k-Means uses the Euclidean distance.

3. (*recalculation*) Recalculate the new prototypes $c_1, \ldots, c_k$ by computing the means of all the assigned vectors.

4. Stop if the prototypes did not change more than a predefined threshold or when a maximum number of iterations has been reached, otherwise go back to step 2.

In our SSC scheme, the set of vectors D to be clustered is the subsequences set $D(X, w)$, where X is a time series and w the sliding window's length. In the assignment step, we employ the *Snapping* distance defined in Equation 6.1. Moreover, we force the initialization step to choose the random subsequences such that they do not overlap in the original time series. Figure 6.5 illustrates the intuition behind the *Snapping* distance measure in the context of k-Means clustering. In the next section, we evaluate this SSC scheme on the InfraWatch strain data.

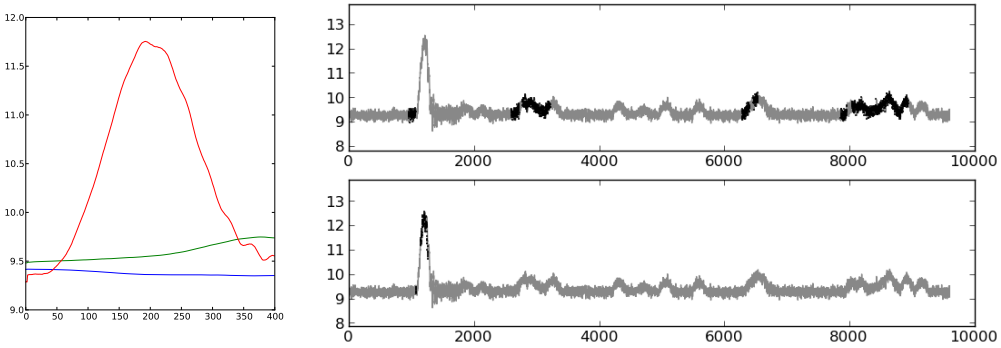


Figure 6.6: Improved results using the *Snapping* distance (see Figure 6.4).

6.4 Experimental Evaluation

In this section, we introduce the experimental setting and we discuss the results of applying the SSC scheme defined in Section 6.3.3 to the strain data.

We considered the following strain time series: `100Seconds` has been collected during the night in a period of low traffic activity across the Hollandse Brug, and consists of 1 minute and 40 seconds of strain data sampled at 100 Hz. The series contains clear traffic events and does not present relevant drift in the strain level due to the short time span. A more substantial series, `FullWeekDay`, consists of 24 hours of strain measurements sampled at 100 Hz, corresponding to approximately 9 millions values. The data has been collected on Monday 1st of December 2008, a day in which the Hollandse Brug was fully operational. All the traffic events expected in a typical weekday, ranging from periods of low activity to congestion due to traffic jams, are present in the data. The temperature throughout the chosen day varied between 4.9 and 7.7 degrees. Figure 6.2 shows an overall plot of the data.

In order to run the defined k-Means SSC scheme, we need to fix a number of parameters. The window length w has been chosen to take into account the structural configuration of the bridge and the sensor network. Considering the span in question is 50 meters long, and a maximum speed of 100 km/h,

a typical vehicle takes in the order of 2.5 seconds to cross the span. In order to capture such events, and include some data before and after the actual event, the window length was set to 400, which corresponds to 4 seconds. The number of clusters k directly affects how the resulting prototypes capture the variability in the data. For the `100Seconds` data, we found $k = 3$ a reasonable choice because, considering its short duration, the time series does not present drift in the strain baseline and the variability in the data can be approximated by assuming three kind of events: no traffic activity (baseline) and light and heavy passing vehicles. On the other hand, the `FullWeekDay` data presents much more variability, mostly due to the drift in the measurements which vertically translates all the events to different levels depending on the external temperature. Moreover, traffic jams cause underlying variability in the data. In the `FullWeekDay`, we found $k = 10$ to be large enough to account for most of the interesting, from an SHM point of view, variations in the time series, though we will also show the result with $k = 4$ for comparison. The f parameter affects the size of the neighborhood of subsequences considered by the *Snapping* distance. As the neighborhood gets smaller, the *Snapping* distance converges to the Euclidean. A big neighborhood, on the other hand, could include subsequences pertaining to other events. We experimented with $f = 0.25$, $f = 0.5$ and $f = 0.75$, yelding comparable outcomes. The presented results were all computed using $f = 0.5$. The shift steps parameter imposes a limitation on the number of Euclidean distances to compute for each comparison of a subsequence with a centroid; we fix it to $s = 10$.

6.4.1 Results

Given the chosen parameters, we run both the classic k-Means SSC and the *Snapping* distance variant on the `100Seconds` and `FullWeekDay` data.

Figure 6.6 depicts the results obtained by applying the k-Means SSC based on the *Snapping* distance on the `100Seconds` data. Comparing this with the results using the Euclidean distance on the same data in Figure 6.4, in this case, the big bump-shaped peak, caused by a heavy passing vehicle, is represented by a single prototype, while the remaining prototypes model

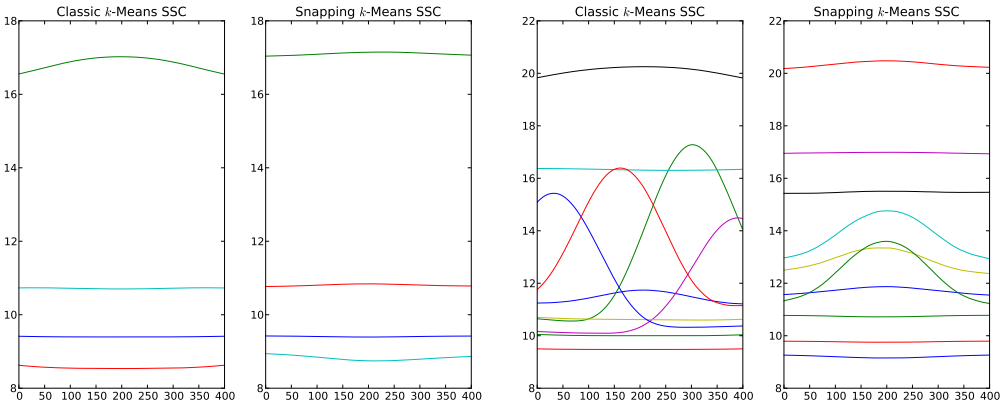


Figure 6.7: Prototypes produced by applying k-Means respectively with Euclidean and Snapping distance on the `FullWeekDay` data, for both $k = 4$ (left) and $k = 10$ (right).

lighter passing vehicles and the strain baseline (whose assignments are not shown in the picture).

Figure 6.7 shows the resulting prototypes obtained from the `FullWeekDay` data for $k = 4$ (left) and $k = 10$ (right). The prototypes computed for $k = 4$ by both the classic and revised k-Means SSC are really similar. Setting $k = 4$ does not account for all the variability in the `FullWeekDay` data and the resulting prototypes try to represent the different strain levels more than the actual events. In this case, the effect of considering the neighborhood of each subsequence, as done by the *Snapping* distance, is dominated by the presence of large differences in the strain values.

The prototypes for $k = 10$ better describe the variability in the data and represent both the different strain levels as well as the individual events (peaks). In this case, the classic k-Means SSC introduces double representations of the same logical events. This is avoided in our revised solution, thus better representing the variability in the data: every prototype now models a different strain level or event, as shown in Figure 6.7 (right).

Although Figure 6.7 gives an idea of the differences between the prototypes produced by the classic k-Means SSC and the *Snapping* version, it does not

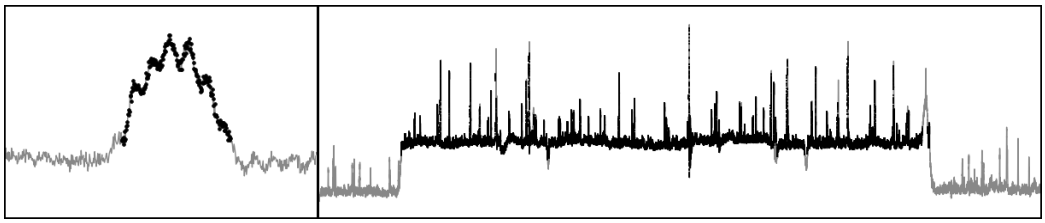


Figure 6.8: Two examples of events represented by individual prototypes. The central point of an associated subsequence is drawn in black.

show how the data is subdivided across them. Figure 6.8 shows two examples, at different time scales, of events associated to a single prototype. The plot on the left shows a heavy passing vehicle (in black), while the plot on the right shows all the subsequences considered part of a traffic jam event.

6.4.2 A Scalable Implementation

Given the amount of data generated by the sensor network, it is important to have a very scalable implementation of our clustering method. Therefore, we have developed a parallelized version based on the MapReduce framework using Hadoop [101]. Indeed, the main bottleneck in clustering lies in calculating the (snapping) distances between every subsequence and the cluster centers, which need to be read from disk. With MapReduce, we can distribute the data reads over a cluster of machines.

An overview of the resulting system is shown in Figure 6.9. In the first stage, we 'massage' the data to prepare it for the clustering phase. Since the computing nodes work independently, they need to be passed complete subsequences, including the lead-in and lead-out, in single records. First, we read the measurements of a single sensor for every timestamp, and its value is *mapped* to the initial timestamp ts of every subsequence in which it occurs. Then, all measurements for a specific ts are *reduced* to a complete subsequence.

In the clustering phase, we first select k random centroids. Then, each subsequence is mapped to the nearest centroid, using the snapping distance,

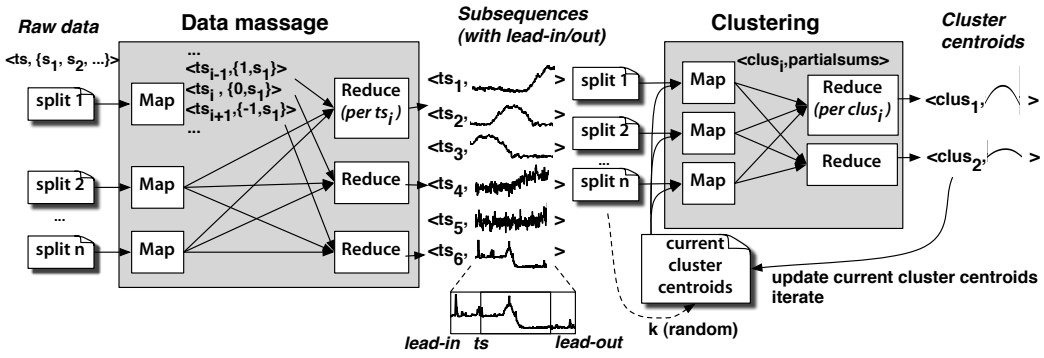


Figure 6.9: MapReduce implementation of our clustering method. Every map or reduce task can be run on any available computing core.

together with the combined points mapped by the same mapper. The reducer receives all points mapped to a certain cluster and calculates the new cluster centroid. This is repeated n times or until the clusters converge. The k-Means implementation is an adapted version of k-Means found in the Mahout library.[1]

We evaluated this implementation on a relatively small cluster of 5 quad-core computing nodes. Compared to a sequential implementation that loaded all the `FullWeekDay` data in memory, it yielded a 6-fold speedup in spite of the extra I/O overhead. Moreover, it scales linearly, even slightly sub-linearly, to time series of several months. For example, one month of data was clustered (using 10 iterations) in less than 14 hours, and can be sped up further by simply adding more nodes.

6.5 Conclusion

In this chapter, we have focused on the problem of identifying traffic activity events in strain measurements, produced by a sensor network deployed on a highway bridge. Characterizing the bridge's response to various traffic events represents an important step in the design of a complete SHM solution, as it will permit implementations of real-time classification or anomaly discovery

[1]Mahout — Scalable Machine Learning Library. `http://mahout.apache.org`

techniques.

The proposed solution is based on subsequence clustering, a technique shown to be prone to undesired behaviors and whose outcome is strongly dependent on the kind of data it is applied to. In view of this, we studied SSC in relation to the features of the strain data, showing that only some of the documented pitfalls (i.e., multiple representations) occur in our case. To solve this, we introduced a context-aware distance measure between subsequences, which also takes the local neighborhood of a subsequence into account. Employing this *Snapping* distance measure, we showed that SSC by k-Means returns a correct modeling of the traffic events.

Chapter 7

Interactive Time-Series Visualization

When approaching a new data science problem, it is most of the time preferable to spend some time to get an idea of the properties and the features of the data at hand. This exploratory phase is not only useful to get a better understanding of the application domain, but it can provide fundamental insight about the data and inform all the subsequent modeling, feature construction and algorithm design choices. Moreover, freely looking at the data before diving into the actual modeling could spot fundamental issues on how it was collected and processed in the first stage, issues which could potentially render any derived analysis flawed if not pointless.

This preliminary exploration phase of a given dataset is widely known in the community under the name of *exploratory data analysis* (EDA) [92, 93]. Typical EDA activities include generating statistical summaries, computing aggregations, fitting distributions and, last but not least, visualizing the data in several ways in order to spot patterns and gain insight, mostly driven by the intuition of the practitioner.

Effective visualization, in fact, is one of the most powerful and immediate ways of analyzing a dataset, relying on the ability of the human brain to abstract, summarize, spot trends and anomalies through visual inspec-

tion [46].

Most data analysis software suites allow practitioners to perform visualization easily on moderately sized datasets. However, when the data at hand is too big to be handled by off-the-shelf software solutions, it becomes difficult to perform effective visualization and more advanced solutions, able to cope with big data, are required.

This chapter discusses visualization in the context of EDA of massive time series data. In particular we focus on the following problem:

> Given a massive time series dataset, how can we support interactive visualization, enabling fast browsing and zooming from coarser to finer levels of detail?

Note how the emphasis is put on the interactivity of the process as required by effective EDA. Although it is reasonably easy to visualize small datasets with current software suites, this simple task becomes quite challenging when dealing with large amount of data, especially when it is required to do this interactively. Throughout the chapter, we will see how this problem can be addressed by working on sampled versions of the original data, organized as a hierarchy.

The solution has been implemented and tested on a real-world scenario, the InfraWatch project, and resulted in a software package called VizTool, which will be introduced throughout the chapter.

The rest of the chapter is organized as follows. Section 7.1 introduces the concept of sub-sampling hierarchy which lays the foundation for the subsequent sections. Section 7.2 discusses how we exploit the sub-sampling hierarchy in order to support interactive visualization of massive time series data. Section 7.3 introduces VizTool, a web application implementing the concepts discussed in the chapter. Finally, Section 7.4 draws the conclusions.

7.1 Hierarchical Time Series Subsampling

When dealing with large univariate time series, there are mainly two properties that affect dimensionality: the extent of the measured period and its sampling frequency. By reducing the considered period, the sampling frequency, or both, one can directly reduce the amount of data to be processed at once, in some cases without losing relevant information for the task at hand.

Take, for example, the case of InfraWatch where data from sensors is sampled at 100 Hz. If we are interested in spotting seasonal trends in strain measurements, it suffices to consider a moderately sub-sampled version of the data, possibly containing one measurement every hour or, even, every day.

In general this means that, depending on the task at hand, it is often enough to consider a sub-sampled version of the data in order to speed up calculations and support interactivity if required.

In this section, we propose a storage scheme for large time series based on progressively sub-sampled versions of the original data to support this idea.

7.1.1 Sub-sampling Hierarchy Construction

The problem of effectively reducing the dimensionality of a time series, interpreted as its number of data points, boils down to producing a reduced representation such that it resembles the original data as much as possible. From now on, and without loss of generality, we will assume we are dealing with constant rate time series[1].

The most trivial way to reduce the dimensionality of such a time series is to consider every nth point, thus reducing its size by a factor n. This approach, however, has a drawback as it is too sensitive to outliers. In fact, rare events, such as spikes in the data or errors in the measurements, could

[1] It is always possible to add or remove points of a time series, by interpolation, to make its sampling rate constant.

be selected in the sampling, rendering the resulting approximation, and its shape, skewed.

A more robust way of producing a low dimensionality approximation is by taking the average of the points to be aggregated. More formally, given a time series $\mathbf{x}$ of length n, we want to compute an approximation of length M

$$A(\mathbf{x}, M) = \hat{\mathbf{x}} = \hat{x}[1], \ldots, \hat{x}[M]$$

where:

$$\hat{x}[i] = \frac{M}{n} \sum_{j=n/M(i-1)+1}^{(n/M)i} x[j] \ .$$

This method of reducing the dimensionality of a time series is also at the base of a well known segmentation technique called Piecewise Aggregate Approximation [47]. In the segmentation task, however, the final outcome is not a time series with a reduced number of points (a lower sampling rate) but a cheaper representation of the same data, obtained by reducing segmented data points to horizontal lines centered around their mean.

The average function, however, is not the only possible choice for aggregating time series data. Ideally, the choice of an aggregation function should be based on its ability to preserve the perceptual features of the data, although this ability may depend on the properties of the data itself and how it evolves over time. A review of several possible aggregation functions and analysis of how well they cope with the task of visualization can be found in [6].

We also note that even basic aggregation functions can be of practical importance when dealing with time series data, especially for visualization purposes. Consider, for example, the case of temperature sensors where one is often interested in knowing what are the changes in temperature at any given time frame. In such cases, computing multiple types of aggregations, such as minimum, maximum and average, can support *band visualizations*.

Having a way to reduce the length of a time series, we can now define a hierarchical storage scheme which materializes different levels of approximation given an input time series.

Definition (Hierarchical Storage Scheme). Given a time series $\mathbf{x}$ of length n, a hierarchical storage scheme is defined by materializing a sequence of $log_2 n + 1$ consecutively coarser approximations $A(\mathbf{x}, n/2^i)$, for $0 \leq i \leq log_2 n$.

Note that we apply the floor operation on $log_2 n$ to handle time lengths that are not strictly a power of two.

In other words, we store consecutively coarser versions of the original time series, where every version has half the size of the immediately finer one. This approach creates a pyramid of approximations which, when fully stored on disk, permits to quickly retrieve portions of a time series at a resolution that is as close as possible to the desired one. Figure 7.1 depicts this concept visually.

We note that, for any given time series $\mathbf{x}$ of length n, the total number of data points of all levels of approximation in the storage scheme is $2n$. As the storage capacity quickly increases over time and its price quickly drops, a twofold increase of the required storage represents a good compromise, especially considering the huge gain in retrieval speed provided by this approach for visualization purposes.

The concept of considering data stored at different resolutions and complexities in order to speed up retrieval and rendering is also at the base of many computer graphics methods. In texture mapping [39], for example, a technique called mipmapping [102] involves pre-calculating sequences of progressively lower resolution versions of the same texture image, each of which is half the size of the previous one. The approach permits to reduce render time and reduce artefacts, such as aliasing, by choosing the right level of resolution depending on the pixel density of the object. Objects closer to the camera will be rendered with high resolution textures while, on the contrary, for distant objects, using less-defined texture images will suffice.

The same concept, again in computer graphics, is behind LOD (level of detail) based 3D rendering [66], where the polygonal complexity of an object is lowered, by employing polygonal reduction algorithms [65], as it moves away from camera. Although more advanced continuous LOD methods exist, basic (discrete) LOD approaches are based on storing pre-computed polygonal

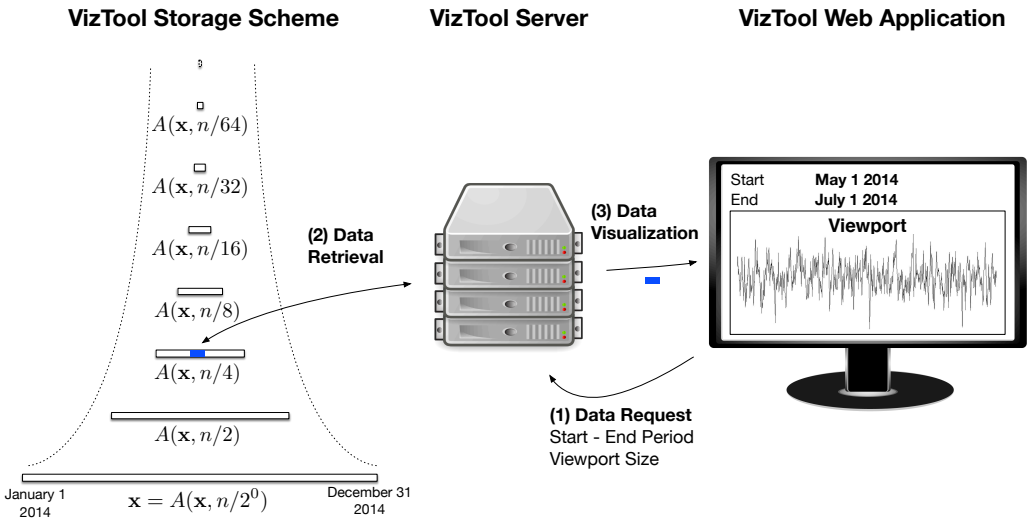


Figure 7.1: The three main components of VizTool. On the left, it is shown the pyramidal storage scheme for time series introduced in 7.1.1. A schematic example of a VizTool user session is shown on the right side. The web application makes a data retrieval request to the VizTool server providing the desired period and the size of the viewport. Given these parameters, the server accesses the best resolution level from the storage and sends the approximated data back to the web application for visualization.

representations of objects at different resolutions.

7.2 Interactive Visualization

We will now show how the time series storage scheme introduced in Section 7.1 can be leveraged to support interactive visualization of large time series.

First, let us consider a typical time series visualization scenario. The following parameters affects the produced visualization at a given time:

- the requested time period (t_{start}, t_{end}),
- the sampling rate of the original data,

- the width in pixels of the visualization viewport w.

Any given configuration of time period and sampling rate results in a number of data points, from the original time series, to be visualized. As this number of points can potentially exceed the pixel width of the viewport, some sort of aggregation has to take place. Performing such an aggregation task on the fly, starting from the original time series, can be an expensive process and could severely harm the interactivity of the visualization.

In order to speed up the process above, we can employ the storage scheme introduced in Section 7.1.1 to directly retrieve the data from the most suitable aggregation level, without performing on the fly expensive operations.

Ideally, given a configuration of time period and sampling rate, we would like to retrieve a number of points that takes into account the size of the viewport in order to reduce the amount of data to be aggregated at render time.

Given a time series $\mathbf{x}[t_{start}, t_{end}]$ to be visualized, let $R = (t_{end} - t_{start})/w$ be the ratio between the requested number of points in the original time series and the viewport width. The value of R indicates how many data points per pixel are to be shown using the original time series. We can reduce this factor by retrieving the data from the right approximation level in the storage scheme. More formally, the best option is to retrieve the data at level $A(\mathbf{x}, n/2^k)$ where $k = \lfloor log_2 R \rfloor$.

This approach links the size of selected data to the resolution of the screen by retrieving the largest aggregation level that has enough data points for the current viewport. One clear advantage is that data retrieval time is reduced as the complexity of the visualization is now dependent on the actual resolution of the screen used.

7.3 VizTool Software

The storage scheme and the interactive visualization method introduced in the previous sections are the core concepts behind VizTool, a visualization

software for large time series data. VizTool has been developed in the context of the InfraWatch project to aid interactive visualization of the bridge's data and facilitate the discussions with the domain experts.

The software has been designed with the following goals in mind:

- fast and interactive visualization of large time series data collected from sensors,
- ability to adapt the details in the visualization to the actual viewport size,
- possibility of comparing data from multiple sensors by stacking multiple time series line charts,
- support for band visualization based on minimum, maximum and average aggregation functions,
- ability to export any portion of the data at any given sampling rate,
- ability to bookmark and add notes to portions of the data to support data annotation activities,
- possibility to compute correlations between any chosen set of sensors.

The second goal, in particular, was fundamental to run VizTool effectively on large monitors or multi-screen setup when discussing data and presenting projects results.

Moreover, because of the sensitiveness and size of InfraWatch's data, VizTool's architecture is based on a client-server model which permits to host all the data on the server side, while allowing client hosts to browse and export portions of it.

VizTool's server-side application has been developed in Python [77] using the web framework Django [23] and HDF5 [38] as data storage library to support effective and efficient caching of data from disk into memory. On the client-side, the web application is written in pure Javascript and HTML/CSS employing the Highcharts [40] graphing library for plotting the time series.

VizTool has been used to visualize InfraWatch's data at scale, allowing one

Figure 7.2: The VizTool web application interface.

to browse and inspect a terabyte sized dataset of sensor time series sampled at 100 Hz for a period of three months.

An example of a VizTool session is shown in Figure 7.2. Figure 7.3 shows VizTool running on a multi-screen setup.

7.4 Conclusions

In this chapter, we introduced the task of large time series visualization in the context of Exploratory Data Analysis (EDA) and we discussed the challenges linked to effective interactive visualization of big data.

As we observed that the amount of visible data points is always limited by

the size of the visualization viewport, we proposed a storage scheme to hold sub-sampled versions of the original data in order to speed up data retrieval at different levels of resolution.

We introduced a data retrieval mechanism for visualization based on such storage scheme and presented VizTool, a software solution that leverages these concepts to support fast and interactive visualization of large time series data collected from sensors.

VizTool proved to be an effective tool for the practical exploration of the InfraWatch data, not only supporting the EDA process but also serving as a practical demonstration for public exposition of the project's results.

VizTool was also instrumental in discovering important properties and events in the data such as dead sensors, re-calibration activities, correlation between temperature and strain response, differences of traffic activity between work days and weekend days. Moreover, VizTool made even more evident the multi-scale nature of InfraWatch data and how different scales (monthly, daily, hourly) reveal new complex phenomena and events inherent to certain time resolutions.

Future work includes the extension of the VizTool software to support on the fly operations that leverage the storage scheme we presented. For example, it would be possible to compute approximate correlations between time series in an anytime fashion by starting the computation at the coarsest level, progressively refining the results while moving to finer levels.

The same concept could be applied to motif discovery. Anytime approximate motif discovery could be implemented by exploiting the sub-sampling hierarchy. A naive solution would involve running the exact MK algorithm [74] at each level of the hierarchy, from coarser to finer resolutions, and presenting intermediate results to the user.

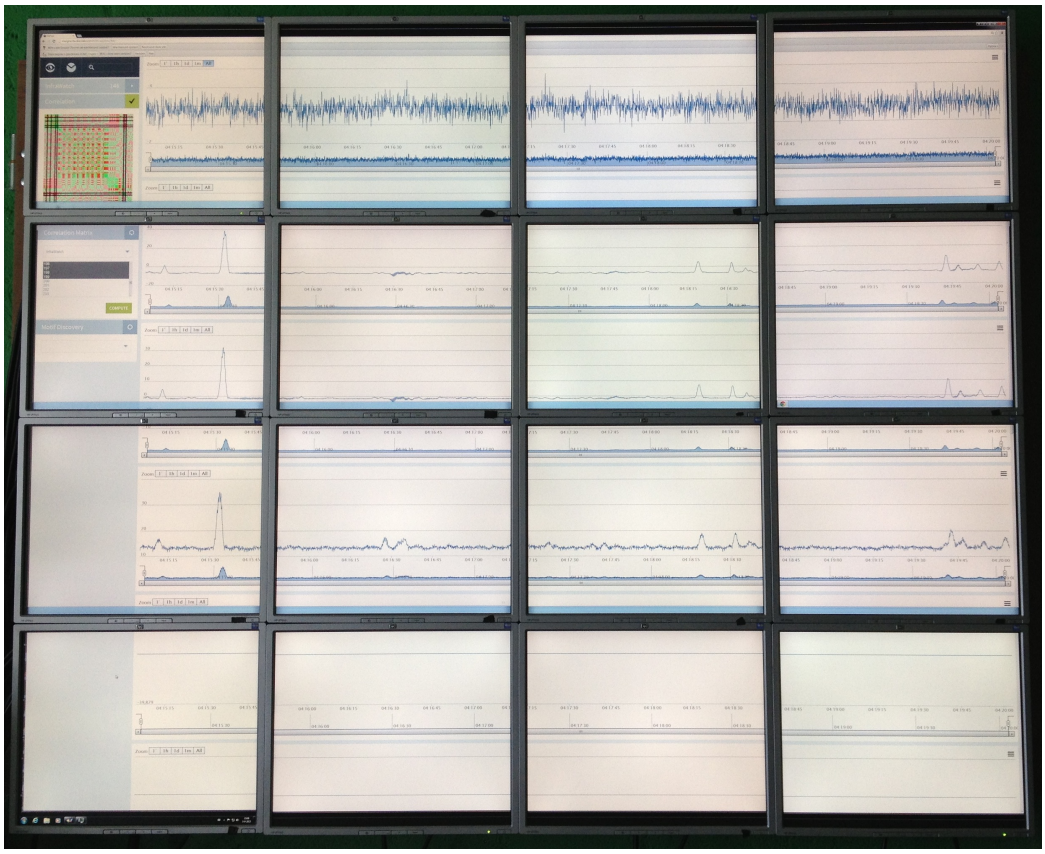


Figure 7.3: VizTool running on a multi-screen setup at the Leiden Institute for Advanced Computer Science. The setup mounted a total of 16 screens (4 by 4) for a total resolution of 5120x4096.

Chapter 8

Conclusions

In this thesis, we discussed data mining methods and algorithms for the unsupervised analysis of time series sensor data from complex physical systems. As complex systems are often affected by several phenomena at once, we developed techniques able to cope with these complexities, such as the presence of noisy measurements, multiple temporal scales and recurring patterns at play at the same time. We also presented a technique and a software tool to make the interactive visualization of massive time series datasets feasible.

We summarize our contributions below.

- In Chapter 4, we introduced a data mining method to discover the relevant temporal scales in a time series. We employ the scale-space theory and the Minimum Description Length principle to select the most relevant time series decomposition among the many possible by sub-diving the scale-space image. We introduced two different encoding schemes aimed at exploiting (making possible to compress) different properties of the data. We have shown how the presented methodology gives meaningful results in both artificial and real-world scenarios.
- A byproduct of the method is that it produces an optimal decomposition such that each component is represented according to its inherent complexity and does not present interferences from phenomena at other

temporal scales. We have shown an example of how these individual per-component representations may better serve tasks like classification, regression or association analysis than the input, mixed, time series.

- In Chapter 5, we introduced a method for the discovery of multi-scale motifs in time series data. Another typical property of complex systems is the presence of recurring phenomena at different temporal scales. Moreover, these recurring phenomena appear warped in time, more or less intense or both throughout the data. We employ the scale-space theory and Minimum Description Length as the base of our methodology. Differently from much of the existing literature, we employ a definition of motifs based on structural similarity, other than one based on point-wise comparisons, in order to account for warping and deformations. We propose a way to transform the structural representation of a motif into a symbolic string which allows for fast matching by means of suffix arrays. The effectiveness of the method is proved on sensor data from several applications, including InfraWatch data.

- In Chapter 6, we focused on the problem of identifying traffic activity events in strain measurements. The proposed solution is based on subsequence clustering, a technique known to be prone to undesired behaviors and whose outcome is strongly dependent on the kind of data it is applied to. In view of this, we studied SSC in relation to the features of the strain data, showing that only some of the documented pitfalls (i.e., multiple representations) occur in our case. To solve this, we introduced a context-aware distance measure between subsequences, which also takes the local neighborhood of a subsequence into account. Employing this *Snapping* distance measure, we showed that SSC by k-Means returns a correct modeling of the traffic events.

- In Chapter 7, we introduced the task of large time series visualization in the context of Exploratory Data Analysis (EDA). We proposed a storage scheme to hold sub-sampled versions of the original data in order to speed up data retrieval at different levels of resolution. Based on this storage technique, we presented a data retrieval mechanism

for visualization and VizTool, a software solution to support fast and interactive visualization of large time series data collected from sensors. VizTool was instrumental in discovering important properties and events in real-world sensor data such as InfraWatch, where it was helpful to discover dead sensors, re-calibration activities, correlation between temperature and strain response, differences of traffic activity between work days and weekend days.

One of the main themes of the thesis is how complex systems often exhibit diverse behaviors at different temporal scales. A general conclusion is that data mining methods should be able to cope with the multiple resolutions (scales) at the same time in order to fully understand the data at hand and extract useful information from it. This becomes more and more important as we advance in our ability to collect increasingly detailed data about the phenomena around us.

Throughout the thesis, we developed data mining methods aimed at coping with this challenge. An important conclusion is that the Minimum Description Length principle, paired with the scale-space construction, represents an effective way of discerning what is fundamental and what is not in the data while considering different scales of analysis. This same principle was instrumental in developing effective techniques to both detect the relevant scales and mine the patterns that occur.

The importance of looking at different scales is also connected to visualization. Complex systems show different phenomena at different resolution and interactive visualization can effectively help practitioners to focus on a specific resolution without being overwhelmed by the finer details.

One nice property of the methods we developed is that they do not require parameters. This is a result of our choice of employing the MDL principle and a model selection approach. However, we note that, in order for this approach to be effective in practice, it is important to define flexible encoding schemes that are able to capture the variability of the data at hand, possibly incorporating domain knowledge.

8.1 Future Work

Future work includes extending the presented methods to work in a streaming context. This would allow, for example, to discover new phenomena in a quasi-real time fashion or detect the presence of novel recurring patterns.

Another promising opportunity for future work is to explore how MDL and the scale-space theory could be used to design anomaly detection techniques at multiple scales. Anomaly detection is, in fact, an important task when dealing with the monitoring of complex systems, and MDL, other than as a model selection principle, could be employed to spot changes in the underlying data generation processes.

Finally, a relevant task linked to the goals of InfraWatch is the analysis of multivariate time series data in order to mine key performance indicators (KPIs) that behave in an approximately monotonic way. In fact, when looking for indicators such as the degradation of concrete structures, as in the case in InfraWatch, one approach is to look for subsets of sensors that, when combined through linear and non-linear operators, result in a monotonic time series. Such derived KPIs are clearly indicators of some underlying process that irreversibly moves in a certain direction, and degradation of the bridge is a good candidate for that. Using an approach called Equation Discovery [24], one might discover such monotonic functions.

Bibliography

[1] C. C. Aggarwal. *Managing and mining sensor data*. Springer Science & Business Media, 2013.

[2] K. Aki and P. G. Richards. *Quantitative Seismology*. University Science Books, 2 edition, 2002.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *Communications magazine, IEEE*, 40(8):102–114, 2002.

[4] K. Ashton. That internet of things thing. *RFiD Journal*, 22(7):97–114, 2009.

[5] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805, 2010.

[6] A. Baggio, U. Vespier, and A. Knobbe. Selection of Data Adaptive Approximations for Large Time Series Visualization. In *Proceedings of Benelearn 2013*, 2013.

[7] H. Bai, M. Atiquzzaman, and D. Lilja. Wireless sensor network for aircraft health monitoring. In *Broadband Networks, 2004. BroadNets 2004. Proceedings. First International Conference on*, pages 748–750. IEEE, 2004.

[8] S. Barker, A. Mishra, D. Irwin, E. Cecchet, P. Shenoy, and J. Albrecht. Smart*: An open data set and tools for enabling research in sustainable homes. *SustKDD, August*, 2012.

[9] T. Becker, M. Kluge, J. Schalk, K. Tiplady, C. Paget, U. Hilleringmann, and T. Otterpohl. Autonomous sensor nodes for aircraft structural health monitoring. *Sensors Journal, IEEE*, 9(11):1589–1595, 2009.

[10] R. Bertens and A. Siebes. Characterising seismic data. In *Proceedings of the 2014 SIAM International Conference on Data Mining, Philadelphia, Pennsylvania, USA, April 24-26, 2014*, pages 884–892, 2014.

[11] M. A. Beyer and D. Laney. The importance of 'big data': A definition. Technical report, Gartner, 2012.

[12] R. Cachucho, M. Meeng, U. Vespier, S. Nijssen, and A. Knobbe. Mining multivariate time series with mixed sampling rates. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 413–423. ACM, 2014.

[13] J. R. Chen. Making clustering in delay-vector space meaningful. *Knowl. Inf. Syst.*, 11(3):369–385, April 2007.

[14] M. Chen, S. Gonzalez, A. Vasilakos, H. Cao, and V. C. Leung. Body area networks: A survey. *Mobile networks and applications*, 16(2):171–193, 2011.

[15] M. Chen, S. Mao, Y. Zhang, and V. C. Leung. Big data applications. In *Big Data*, pages 59–79. Springer, 2014.

[16] C.-Y. Chong and S. P. Kumar. Sensor networks: evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256, 2003.

[17] R. Cilibrasi and P. Vitanyi. Clustering by compression. *Information Theory, IEEE Transactions on*, 51(4):1523–1545, 2005.

[18] P. Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3):287–314, April 1994.

[19] R. A. Davis, T. C. M. Lee, and G. A. Rodriguez-Yam. Structural break estimation for nonstationary time series models. *Journal of the American Statistical Association*, 101(473):223–239, 2006.

[20] H. de Vries, G. Azzopardi, A. Koelewijn, and A. Knobbe. Parametric Nonlinear Regression Models for Dike Monitoring Systems. In *Proceedings of IDA '14*, 2014.

[21] M. Dejori, H. H. Malik, F. Moerchen, N. C. Tas, and C. Neubauer. Development of data infrastructure for the long term bridge performance program. In *Proceedings of Structures*, volume 9, 2009.

[22] A. M. Denton, C. A. Besemann, and D. H. Dorr. Pattern-based time-series subsequence clustering using radial distribution functions. *Knowl. Inf. Syst.*, 18(1):1–27, January 2009.

[23] Django. Django Web Framework. `https://www.djangoproject.com`. Accessed: 2015-06-12.

[24] S. Dzeroski and L. Todorovski. Discovering dynamics: From inductive logic programming to machine discovery. *Journal of Intelligent Information Systems*, 4:89–108, 1995.

[25] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan. The pothole patrol: using a mobile sensor network for road surface monitoring. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pages 29–39. ACM, 2008.

[26] C. Farrar, S. Doebling, and D. Nix. Vibration-based structural damage identification. *Philosophical Transactions of the Royal Society: Mathematical, Physical & Engineering Sciences*, 359:131–149, 2001.

[27] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996.

[28] FHWA. Reliability of visual inspection for highway bridges. Technical Report FHWA-RD-01-020, Federal Highway Administration Report, 2001.

[29] E. Frank, C. Chui, and I. H. Witten. Text categorization using compression models. In *Proceedings of the Conference on Data Compression*, pages 555–, Washington, DC, USA, 2000. IEEE Computer Society.

[30] R. Fujimaki, S. Hirose, and T. Nakata. Theoretical analysis of subsequence time-series clustering from a frequency-analysis viewpoint. In *Proceedings of SDM 2008*, pages 506–517, 2008.

[31] J. Gama. *Knowledge discovery from data streams*. CRC Press, 2010.

[32] R. Ganti, I. Mohomed, R. Raghavendra, and A. Ranganathan. Analysis of data from a taxi cab participatory sensor network. In *Mobile and ubiquitous systems: Computing, networking, and services*, pages 197–208. Springer, 2012.

[33] J. Gantz and D. Reinsel. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Analyze the Future*, 2007:1–16, 2012.

[34] P. D. Grünwald. *The Minimum Description Length Principle*. The MIT Press, 2007.

[35] S. Gusmeroli, S. Haller, M. Harrison, K. Kalaboukas, M. Tomasella, O. Vermesan, H. Vogt, and K. Wouters. Vision and challenges for realising the internet of things. *status: published*, 2010.

[36] J. Hagenauer, Z. Dawy, B. Goebel, P. Hanus, and J. Mueller. Genomic analysis using methods from information theory. In *Information Theory Workshop, 2004. IEEE*, pages 55–59. IEEE, 2004.

[37] J. Han, M. Kamber, and J. Pei. *Data mining: concepts and techniques: concepts and techniques*. Elsevier, 2011.

[38] HDF Group. HDF5 Library. `https://www.hdfgroup.org/HDF5`. Accessed: 2015-06-12.

[39] P. S. Heckbert. Survey of texture mapping. *Computer Graphics and Applications, IEEE*, 6(11):56–67, 1986.

[40] Highcharts. Highcharts Charting Library. `http://www.highcharts.com`. Accessed: 2015-06-12.

[41] F. Höppner. Time series abstraction methods - a survey. In *Informatik bewegt: Informatik 2002 - 32. Jahrestagung der Gesellschaft für Informatik e.v. (GI)*, pages 777–786. GI, 2002.

[42] B. Hu, T. Rakthanmanon, Y. Hao, S. Evans, S. Lonardi, and E. Keogh. Discovering the intrinsic cardinality and dimensionality of time series using mdl. In *Proceedings of ICDM 2011*, pages 1086–1091, 2011.

[43] T. Idé. Why does subsequence time-series clustering produce sine waves? In *Proceedings of ECML PKDD 2006*, pages 211–222, 2006.

[44] K. Judd and A. Mees. On selecting models for nonlinear time series. *Physica D*, 82(4):426–444, May 1995.

[45] H. Kargupta, R. Bhargava, K. Liu, M. Powers, P. Blair, S. Bushra, J. Dull, K. Sarkar, M. Klein, M. Vasa, et al. Vedas: A mobile and distributed data stream mining system for real-time vehicle monitoring. In *SDM*, pages 300–311, 2004.

[46] D. A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–8, 2002.

[47] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*, 3(3):263–286, 2001.

[48] E. Keogh, S. Chu, D. Hart, and M. Pazzani. Segmenting time series: A survey and novel approach. In *In an Edited Volume, Data mining in Time Series Databases. Published by World Scientific*, pages 1–22. Publishing Company, 1993.

[49] E. Keogh and J. Lin. Clustering of time-series subsequences is meaningless: implications for previous and future research. *Knowl. Inf. Syst.*, 8(2):154–177, August 2005.

[50] E. Keogh, S. Lonardi, and B. Y.-c. Chiu. Finding surprising patterns in a time series database in linear time and space. *Proceedings of KDD '02*, page 550, 2002.

[51] E. Keogh, S. Lonardi, and C. A. Ratanamahatana. Towards parameter-free data mining. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 206–215. ACM, 2004.

[52] E. Keogh, S. Lonardi, C. A. Ratanamahatana, L. Wei, S.-H. Lee, and J. Handley. Compression-based data mining of sequential data. *Data Mining and Knowledge Discovery*, 14(1):99–129, 2007.

[53] E. J. Keogh, S. Chu, D. Hart, and M. J. Pazzani. An online algorithm for segmenting time series. In *Proceedings of ICDM 2001*, pages 289–296, 2001.

[54] B. Khoo. Rfid- from tracking to the internet of things: A review of developments. In *Proceedings of the 2010 IEEE/ACM Int'L Conference on Green Computing and Communications & Int'L Conference on Cyber, Physical and Social Computing*, GREENCOM-CPSCOM '10, pages 533–538, Washington, DC, USA, 2010. IEEE Computer Society.

[55] A. Knobbe *et al.* InfraWatch: Data management of large systems for monitoring infrastructural performance. In *Proceedings IDA 2010*, pages 91–102, 2010.

[56] J. Kok, A. Knobbe, H. Blockeel, B. Obladen, and E. Koenders. Large Data Stream Processing for Bridge Management Systems. In *Proceedings SMAR 2011*, 2011.

[57] P. Kontkanen and P. Myllymäki. Mdl histogram density estimation. In *International Conference on Artificial Intelligence and Statistics*, pages 219–226, 2007.

[58] C. Laughman, K. Lee, R. Cox, S. Shaw, S. Leeb, L. Norford, and P. Armstrong. Power signature analysis. *Power and Energy Magazine, IEEE*, 1(2):56–63, 2003.

[59] U. Lee, B. Zhou, M. Gerla, E. Magistretti, P. Bellavista, and A. Corradi. Mobeyes: smart mobs for urban monitoring with a vehicular sensor network. *Wireless Communications, IEEE*, 13(5):52–57, 2006.

[60] J. Lin, E. Keogh, and S. Lonardi. Visualizing and Discovering Non-Trivial Patterns In Large Time Series Databases. *Information visualization*, 2005.

[61] J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Min. Knowl. Discov.*, 15(2):107–144, October 2007.

[62] T. Lindeberg. Scale-space for discrete signals. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 12(3):234–254, March 1990.

[63] X. Liu, N. Krahnstoever, T. Yu, and P. Tu. What are customers looking at? In *Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on*, pages 405–410. IEEE, 2007.

[64] S. Lohr. The age of big data. *New York Times*, 11, 2012.

[65] D. P. Luebke. A developer's survey of polygonal simplification algorithms. *Computer Graphics and Applications, IEEE*, 21(3):24–35, 2001.

[66] D. P. Luebke. *Level of detail for 3D graphics*. Morgan Kaufmann, 2003.

[67] S.-M. Mäkelä, S. Järvinen, T. Keränen, M. Lindholm, and E. Vildjiounaite. Shopper behaviour analysis based on 3d situation awareness information. In *Video Analytics for Audience Measurement*, pages 134–145. Springer, 2014.

[68] M. Mampaey, N. Tatti, and J. Vreeken. Tell me what i need to know: succinctly summarizing data with itemsets. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 573–581. ACM, 2011.

[69] V. Megalooikonomou, Q. Wang, G. Li, and C. Faloutsos. A multiresolution symbolic representation of time series. In *Proceedings of ICDE05*, pages 668–679, 2005.

[70] N. Melnikova, G. Shirshov, and V. V. Krzhizhanovskaya. Virtual dike: multiscale simulation of dike stability. *Procedia Computer Science*, 4:791–800, 2011.

[71] S. Miao, U. Vespier, R. Cachucho, M. Meeng, and A. Knobbe. Predefined pattern detection in large time series. *Information Sciences*, 2015.

[72] C. Michael, L. Markus, and R. Roger. The internet of things. Technical report, McKinsey Quarterly, 2010.

[73] A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, and D. Irwin. Private memoirs of a smart meter. In *Proceedings of the 2nd ACM workshop on embedded sensing systems for energy-efficiency in building*, pages 61–66. ACM, 2010.

[74] A. Mueen et al., E. Keogh, Q. Zhu, S. Cash, and B. Westover. Exact discovery of time series motifs. In *Proceedings of SDM '09*, pages 473–484, 2009.

[75] M. Nixon. *Feature extraction & image processing.* Academic Press, 2008.

[76] S. Papadimitriou and P. Yu. Optimal multi-scale patterns in time series streams. In *Proceedings SIGMOD 2006*, pages 647–658. ACM, 2006.

[77] Python. Python Language. `http://www.python.org`. Accessed: 2015-06-12.

[78] T. Rakthanmanon, E. Keogh, S. Lonardi, and S. Evans. Time Series Epenthesis: Clustering Time Series Streams Requires Ignoring Some Data. In *Proceedings of ICDM '11*, 2011.

[79] P. Ranganathan. From microprocessors to nanostores: Rethinking data-centric systems. *Computer*, 44(1):39–48, 2011.

[80] D. Salomon. *A concise introduction to data compression.* Springer Science & Business Media, 2007.

[81] S. Saria, A. Duchi, and D. Koller. Discovering deformable motifs in continuous time series data. In *Proceedings of IJCAI '11*, 2011.

[82] D. Sculley and C. E. Brodley. Compression and machine learning: A new perspective on feature space vectors. In *Data Compression Conference, 2006. DCC 2006. Proceedings*, pages 332–341. IEEE, 2006.

[83] T. Shany, S. J. Redmond, M. R. Narayanan, and N. H. Lovell. Sensors-based wearable systems for monitoring of human movement and falls. *Sensors Journal, IEEE*, 12(3):658–670, 2012.

[84] A. Siebes. MDL in pattern mining a brief introduction to krimp. In *Formal Concept Analysis*, pages 37–43. Springer, 2014.

[85] S. W. Smith. *The Scientist & Engineer's Guide to Digital Signal Processing*. California Technical Pub, 1997.

[86] H. Sohn, C. R. Farrar, F. M. Hemez, D. D. Shunk, D. W. Stinemates, B. R. Nadler, and J. J. Czarnecki. *A review of structural health monitoring literature: 1996–2001*. Los Alamos National Laboratory Los Alamos, NM, 2004.

[87] W. Staszewski, C. Boller, and G. R. Tomlinson. *Health monitoring of aerospace structures: smart sensor technologies and signal processing*. John Wiley & Sons, 2004.

[88] Y. Tanaka, K. Iwamoto, and K. Uehara. Discovery of time-series motif from multi-dimensional data based on mdl principle. *Journal of Machine Learning*, 58(2-3):269–300, February 2005.

[89] N. Tatti and J. Vreeken. The long and the short of it: summarising event sequences with serial episodes. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 462–470. ACM, 2012.

[90] T. Taylor, A. Pradhan, G. Divekar, M. Romoser, J. Muttart, R. Gomez, A. Pollatsek, and D. Fisher. The view from the road: The contribution of on-road glance-monitoring technologies to understanding driver behavior. *Accident Analysis & Prevention*, 58:175–186, 2013.

[91] W. M. Thorburn. Occam's razor. *Mind*, 24(2):287–288, 1915.

[92] J. W. Tukey. The future of data analysis. *Annals of Mathematical Statistics*, 33(1):1–67, March 1962.

[93] J. W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[94] M. van Leeuwen and A. Siebes. Streamkrimp: Detecting change in data streams. In *Proceedings of ECML-PKDD 2008*, pages 672–687, 2008.

[95] J. Vanschoren, U. Vespier, S. Miao, M. Meeng, R. Cachucho, and A. Knobbe. *Big Data Management, Technologies, and Applications*, chapter Large-Scale Sensor Network Analysis: Applications in Structural Health Monitoring. IGI Global, 2014.

[96] U. Vespier, A. Knobbe, S. Nijssen, and J. Vanschoren. MDL-Based Analysis of Time Series at Multiple Time-Scales. In *Proceedings of ECML PKDD '12*, 2012.

[97] U. Vespier, A. Knobbe, and J. Vanschoren. Traffic events modeling for structural health monitoring. In *Proceedings of IDA '11*, 2011.

[98] U. Vespier, S. Nijssen, and A. Knobbe. Mining characteristic multi-scale motifs in sensor-based time series. In *Proceedings of the 22nd ACM international conference on Conference on Information & Knowledge Management*, CIKM '13, pages 2393–2398. ACM, 2013.

[99] P. M. Vitányi. Compression-based similarity. In *Data Compression, Communications and Processing (CCP), 2011 First International Conference on*, pages 111–118. IEEE, 2011.

[100] J. Vreeken, M. Van Leeuwen, and A. Siebes. Krimp: mining itemsets that compress. *Data Mining and Knowledge Discovery*, 23(1):169–214, 2011.

[101] T. White. *Hadoop, The Definite Guide*. O'Reilly, 2009.

[102] L. Williams. Pyramidal parametrics. In *ACM Siggraph Computer Graphics*, volume 17, pages 1–11. ACM, 1983.

[103] A. P. Witkin. Scale-space filtering. In *Proceedings IJCAI 1983*, pages 1019–1022, San Francisco, CA, USA, 1983.

[104] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[105] I. H. Witten, A. Moffat, and T. C. Bell. *Managing gigabytes: compressing and indexing documents and images*. Morgan Kaufmann, 1999.

[106] L. Yang, S.-H. Yang, and L. Plotnick. How the internet of things technology enhances emergency response operations. *Technological Forecasting and Social Change*, 80(9):1854–1867, 2013.

[107] R. K. Yedavalli and R. K. Belapurkar. Application of wireless sensor networks to aircraft control and health management systems. *Journal of Control Theory and Applications*, 9(1):28–33, 2011.

[108] A. Zoha, A. Gluhak, M. A. Imran, and S. Rajasegarar. Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey. *Sensors*, 12(12):16838–16866, 2012.

Nederlandse Samenvatting

Tegenwoordig kan vrijwel alles, van natuurverschijnselen tot fysieke systemen, bemeten worden, terwijl de verkregen informatie verzameld, opgeslagen en geanalyseerd wordt om nieuwe inzichten te verschaffen. De inzet van meetsystemen voor allerlei soorten industriële, commerciële en consumententoepassingen heeft feitelijk verscheidene kansen geschapen voor het analyseren van complexe systemen op een tot op heden ongekend detailniveau.

In dit proefschrift wordt aangetoond dat complexe systemen vaak verschillend gedrag vertonen op verschillende tijdsschalen, en wordt bepleit dat datamining methoden ook op meerdere resoluties (tijdsschalen) zouden moeten werken, om de betreffende data volledig te doorgronden en er zinvolle informatie uit te extraheren.

Onder deze aannames heb ik methods ontwikkeld en geëvalueerd voor datamining en visualisatie van omvangrijke tijdreeksdata over complexe systemen die verzameld wordt door middel van sensoren. Specifiek heb ik oplossingen aangedragen voor drie fundamentele vraagstukken: het detecteren van patronen op meerdere tijdschalen, het herkennen van herhalende gebeurtenissen, en het interactief visualiseren van uitzonderlijke grote tijdreeksdata.

De door mij geïntroduceerde methoden en algoritmen combineren concepten van datamining, signaalverwerking en de informatietheorie. Ik demonstreer hoe verschillende technieken gecombineerd kunnen worden om de uitdagingen aan te pakken die naar voren komen bij het analyseren van tijdreeksdata uit de praktijk. Voorbeelden daarvan zijn de aanwezigheid van ruis in de metingen, het vóórkomen van twijfelachtige en afwijkende gebeurtenissen, en als laatste het risico van *overfitten* van de data met modellen die nauwelijks

nog generaliseren.

Hoewel de behandelde onderzoeksvragen in dit proefschrift een brede toepasbaarheid hebben, worden de voorgesteld oplossingen geëvalueerd op een specifieke toepassing vanuit InfraWatch, een *Structural Health Monitoring* project dat zich richt op het verwerken en analyseren van data geproduceerd door een sensornetwerk op een Nederlandse snelwegbrug.

De door mij ontwikkelde methodes maken het mogelijk om de relevante tijdsschalen te detecteren die gelden in de InfraWatch gegevens (alsook in andere databronnen). Daarnaast worden de verschillende herhalende patronen (zogenaamde *motifs*) ontdekt, en wordt de interactieve visualisatie van terabytes aan tijdreeksdata mogelijk gemaakt.

English Summary

Today, virtually everything, from natural phenomena to complex artificial and physical systems, can be measured and the resulting information collected, stored and analyzed in order to gain new insight. The adoption and deployment of measurement systems for all sorts of industrial, commercial and consumer applications has, in fact, paved the way to important opportunities for analyzing complex systems at a level of detail never experienced before.

In this thesis, I have shown how complex systems often exhibit diverse behavior at different temporal scales, and that data mining methods should be able to cope with the multiple resolutions (scales) at the same time in order to fully understand the data at hand and extract useful information from it.

Under these assumptions, I have designed and evaluated data mining and visualization methods for large time series data collected from complex physical systems by means of sensors. In particular, I have developed solutions to three fundamental problems: the detection of multi-scale patterns, the recognition of recurrent events, and the interactive visualization of massive time series data.

The methods and algorithms I have introduced combine concepts from data mining, signal processing, and information theory. I have shown how to combine different techniques in order to deal with many of the challenges present when analyzing real-world time series data, such as the presence of noisy measurements, the occurrence of spurious and anomalous events and, ultimately, the risk of over-fitting the data with models that would be hardly

general.

Although the research questions addressed in this thesis have a general applicability, I evaluated the proposed solutions on a real-world scenario provided by InfraWatch, a Structural Health Monitoring project centered around the management and analysis of data collected by a large sensor network deployed on a Dutch highway bridge.

The application of the methods I developed permitted the identification of the relevant scales of analysis in the InfraWatch data (and other datasets also), the detection of the different recurring motifs and the visualization of terabytes of time series data interactively.

Curriculum Vitae

Ugo Vespier was born in Lamezia Terme, Italy on February 27, 1985.

In 2003 he received his high school degree from Liceo Scientifico G. Galilei of Lamezia Terme and started his bachelor study in Computer Science at University of Pisa, in Italy. In February 2007 he received his bachelor degree and started a Master in Theoretical Computer Science at University of Pisa, which included periods of study abroad at the Vrije Universiteit of Amsterdam and at the Parc Científíc de Barcelona. Ugo graduated *cum laude* in July 2010 with a thesis on computer vision and data mining entitled "Bringing Order to Social Photo Collections", which won the *Franco Denoth Best Thesis Award* for its contributions to the social Internet.

In October 2010, Ugo obtained a PhD position at the Leiden Institute of Advanced Computer Science (LIACS), Leiden University, the Netherlands, under the supervision of prof. dr. J. N. Kok and dr. A. J. Knobbe. His research focused on the development of data mining algorithms and methods for large-scale sensor data. In particular, he looked at the problems arising when analyzing multi-scale time series, a common yet challenging type of data produced when measuring complex systems. Ugo also worked on big data visualization methods, and he was teaching assistant for the courses of Logic and Artificial Intelligence at LIACS. Ugo has published several papers in peer reviewed conferences and workshops.

As of December 2014, Ugo lives in Florence, Italy and is Co-Founder of a VC-funded group of startup companies operating at the intersection of eCommerce, Advertising and Online Video.