The Pennsylvania State University
The Graduate School
College of Engineering

ROBUST DYNAMICAL MODEL-BASED DATA REPRESENTATIONS AND STRUCTURING OF TIME SERIES DATA FOR IN-SEQUENCE LOCALIZATION

A Dissertation in
Electrical Engineering
by
Emil Laftchiev

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

May 2015

The dissertation of Emil Laftchiev was reviewed and approved* by the following:

Constantino Lagoa
Professor of Electrical Engineering
Committee Co-Chair
Dissertation Co-Advisor

Sean Brennan
Associate Professor of Mechanical and Nuclear Engineering
Committee Co-Chair
Dissertation Co-Advisor

John Doherty
Professor of Electrical Engineering

Minghui Zhu
Assistant Professor of Electrical Engineering

Christopher Rahn
Professor of Mechanical and Nuclear Engineering

Kultegin Aydin
Department Head of Electrical Engineering

*Signatures are on file in the Graduate School.

Abstract

In the modern era there is an unprecedented ability to actuate via an increasingly cheaper array of actuators, and to sense through a growing, increasingly cheaper, array of sensors. The ability to sense in particular represents both the unique opportunity and the unique challenge of our time. It is an opportunity because through the acquired data, it is possible to glean insight into processes that are still too complex to be modeled correctly, and a challenge because the sheer volume of data generated is often overwhelming.

Paradoxically, despite having the capability to collect vast stores of data, we still lack knowledge of how to best analyze and use this data. In fact, multiple review papers have demonstrated that published methods are too application specific and depend too strongly on the data set size to be deployed on the scale necessary to address the problem. Heated debates rage in the scientific literature on how to best reduce the data into meaningful features and how to apply this reduced data to solve real-world problems.

This thesis proposes a novel approach to reducing the dimension of data into meaningful features that preserve the information of the data, but also enable the localization of a small, newly collected, set of data in a previously stored data set. The process of identifying a location within a stored data set is termed in-sequence localization and it is particularly important in applications such as vehicle localization, economic forecasting, energy generation mode selection, and stream health monitoring, where stored data can be used to infer the present and future state of a process.

To enable the process of in-sequence localization, this thesis proposes to model the data using autoregressive models, such that a small number of model coefficients can be used to represent large subsets of data. Then using these autoregressive models, in-sequence localization is performed by comparing new data with the models and determining model feasibility. Emphasizing the fact that this method

is to be used on practical data, the models are determined such that in-sequence localization is robust to the data collection noise typically observed in sensors.

This thesis is developed around the application of vehicle localization. This motivating application arises from the need to augment position estimates of the Global Positioning System in the event of emergencies or signal occlusions. We return repeatedly to this application because of its intuitive nature when conveying concepts throughout the thesis. To demonstrate the broad applicability of the methods, data from the applications of economic forecasting, energy generation mode selection, stream health monitoring, and random data is used to demonstrate algorithms throughout the text.

Table of Contents

List of Figures

List of Tables

List of Symbols

Data Symbols and Constants:

d, $\bar{\mathbf{d}}$ The letter d is reserved for noise-free time-series data. When multiple correlated time series are present, they are denoted by the index subscript γ (see Chapter 6). When a vector of noise-free data is represented by the constant $\bar{\mathbf{d}}$.

x, $\bar{\mathbf{x}}$ The variable x denotes a corrupted data point, for example $x(t) = d(t) + \eta(t)$. A vector of sequential corrupted data points is denoted by $\bar{\mathbf{x}}$.

v The letter v is used to represent the velocity of a moving vehicle in the descriptions of vehicle data collection. The first usage occurs in Fig. 3.1.

s The letter s denotes the distance traveled by a moving vehicle in the aforementioned figures.

p The letter p denotes the current position of a traveling vehicle during localization.

Probabilistic Symbols and Constants:

η In this thesis the letter η is reserved for the additive noise that is corrupting the time-series data.

σ_η The noise in this thesis is typically Gaussian because of the sensors used in vehicle localization. The standard deviation of this noise is defined as σ_η and is given by the sensor manufacturer.

σ_m A standard deviation σ_m indexed by a model number, m, denotes the output standard deviation of a particular model that is being discussed. In the multiple time series cases, the covariance matrix of the models is denotes as Σ_m.

σ_{e_m} When the distribution characterizes the modeling error of the m^{th} model, the standard deviation or the covariance matrix are expressed as σ_{e_m} or Σ_{e_m}, respectively.

$\mathbb{P}(\cdot)$ Given the description of noise, the probability of an event is described by $\mathbb{P}(\cdot)$.

$\Phi_\eta(\cdot)$ The additive sensor noise, $\mathbb{N}(0, \sigma_\eta)$, is described by the cumulative distribution function $\Phi_\eta(\cdot)$.

$\Phi_{e_m}(\cdot)$ The cumulative distribution function of the model output error is expressed as $\Phi_{e_{m_s}}(\cdot)$.

β The Greek letter β is reserved for the addive bias noise introduced in Chapter 4.

ζ In the multiple time-series, when the uncertainty is aggregated on the right-hand side of the equation, this aggregate uncertainty is represented by ζ.

ρ The constant ρ is used to bound the minimum probably of success at a transition point when finding the equivalent deterministic expression for a probabilistic expression.

Modeling Symbols and Constants:

A The capital letter, A, is used to denote the Auto-Regressive Model with an exogenous input (ARX) that describe the data in this thesis. This letter is used both in the single and multiple dimensional cases.

c The coefficients of the models are designated by the letter c.

N The capital letter, N, signifies model order for the ARX models in this thesis. Throughout this document the order five is used for the numerical experiments.

e,$\mathbf{e}$ The letter, e, denotes the modeling error of a given model and is indexed by the model from which it is produced and the time index. Additionally a boulded $\mathbf{e}$ is used to denote a vector of error that is found when evaluating a set of data points simultaneously.

λ In Chapter 5, the variable λ is used to describe the zeros of the system equation.

ε,$\boldsymbol{\varepsilon}$ The Greek letter, ε, is used to denote the modeling error bound and is indexed by the model number. A boulded $\boldsymbol{\varepsilon}$ denotes a vector of modeling error bounds such as those seen in the multiple time series case.

τ_m The Greek letter, τ is used to denote the transition point where one model's segment ends and another model's segment begins. The index m denotes the ending point of the m^{th} model segment.

ω In the matrix model equations, the vector ω contains all data points input into the models. This data is typically noisy.

z The vector that contains the subset of points against which model agreement is tested is represented by z.

N_L In a model structure the capital letter L is used to denote the number levels of the structure.

m, N_{ML} Correspondingly, the number of models per level is denotes as N_{ML}. Each model on a given level is indexed by the index m.

N_s The number of segments for previously published data representations are denoted by the letter N_s.

φ,Ψ The symbols of previous data representations surveyed in this thesis are represented by the Greek letter φ, and the total representation is represented by Ψ.

Indices and Other Variables:

t The index t is used to denote the time instant at which the data is collected.

j The index j is used throughout this thesis as a data index representing the unknown location of a small subset of the data within the larger collected data set.

k The index k is used throughout this thesis as a data index representing the length of a small subset of the data. The first and last data point indices are represented as k_s and k_e, respectively.

i The index s is used as a data index representing the length of the full data set within which localization is occurring.

T The letter T is used to denote the time delay in acquisition of the localization subset of data.

N_{ts} The variable N_{ts} is used to denote the number of time-series used in the multi-attribute time-series localization. The time indices or corresponding model dimensions are indexed by γ.

Mathematical Operations

vector inequality: $\leq$ or $>$ Throughout this dissertation vector inequalities (for two given vectors $\mathbf{a}$ and $\mathbf{b}$) of the form $\mathbf{a} \leq \mathbf{b}$ are used to denote the row-wise inequality of the elements of the two vectors. For example when comparing the modeling error vector to the modeling error bound vector,

$$\mathbf{e}_m \leq \boldsymbol{\varepsilon}_m \Leftrightarrow \begin{pmatrix} e_{m_1} \leq \varepsilon_{m_1} \\ \vdots \\ e_{m_k} \leq \varepsilon_{m_k} \end{pmatrix}.$$

vector inequality: $\gtrless$ — In the case where the symbol $\gtrless$ is used to denote a vector inequality of two vectors $\mathbf{a}$ and $\mathbf{b}$, the specified inequality contains some row-wise inequalities that are less than or equal to the right hand hand term, and some row-wise inequalities that are greater than the right hand term, i.e. using the model error comparison from above $\mathbf{e}_m \gtrless \boldsymbol{\varepsilon}_m \Leftrightarrow \begin{pmatrix} e_{m_1} \leq \varepsilon_{m_1} \\ \vdots \\ e_{m_k} > \varepsilon_{m_k} \end{pmatrix}$.

scalar inequality: $\gtrless$ — In the case of scalar inequalities denoted by $\gtrless$, this symbol refers to the evaluation of whether the magnitude of the left hand side is greater than or less than the right hand size, i.e. we evaluate whether a given row of one vector is greater than or less than the row of another. Extending the modeling error example from above $\|e_{m_k}\| \gtrless \varepsilon_{m_k} \Leftrightarrow \|e_{m_k}\| \leq \varepsilon_{m_k}$ or $\|e_{m_k}\| > \varepsilon_{m_k}$.

Acknowledgments

Its been said before that no thesis has ever been written alone. Anyone who has ever completed a dissertation knows that nothing written in this document can be truer than this statement. I would like to thank my advisors Constantino Lagoa and Sean Brennan for the countless hours of advice both academic and personal that contributed so much to my growth in graduate school. I could not have done this without you.

I would like to thank my committee: John Doherty, Minghui Zhu, and Chris D. Rahn for their collaboration and support.

I would like to thank my lab mates both past and present that helped set the stage for my research. I especially want to thank Kshitij Jerath, who never misses an event and is always ready to help, Pramod Vemulapalli, who continues to offer advice and support long after he has graduated, Ashkan Jasour, for his comical approach to life that made conferences presentations easier.

I would like to thank the Electrical Engineering Department for offering me their support and allowing me to teach while finishing this degree. Thanks to David Salvia, who has been a tremendous mentor for me as a teacher. Thanks to Kultegin Aydin for offering me a hand when I needed it.

I am also deeply indebted to my family for their support. I would like to thank my father, Ivan Laftchiev, for his always timely words of wisdom and encouragement, my mother, Vessela Stefanova, for her undying support and love, my brother, Christian Laftchiev, for his sage advice on life, and Sarah Yoder, who is always patient, loving and understanding.

Finally, this dedication would not be complete without mentioning my friend, Joseph Kasprzyk, a person whose shoulder I have leaned on one too many times, and SherryDawn Jackson, who was my home away from home so often in this process.

Thank you all, you made this work possible.

Dedication

To my family and Joe.

Chapter 1

Introduction

Since the dawn of civilization, progress has been driven by the collection of data that facilitates decision making. In the period known as the scientific Renaissance (after the 17th century), the pace of data collection increased as systematic methods of analysis were developed that required repeated experiments for verification [1]. Today, at the beginning of the third millennium, data collection occurs at a dizzying pace and is increasing with every passing day [2].

The data collected today will enable new applications that change all aspects of science. For example, the field of signal processing in the 20th century has been focused on obtaining the best possible estimate of a signal given a single (or several) time series of data. However, as sensing technology continues to decrease in cost and the volume of collected data increases, the more pressing question has become how to process, store and compress the large tracts of data while preserving the most useful information from them. The information that is retrieved from this data is then used to enable the design of new applications that were previously too complex to model. This new paradigm builds on previous work in signal processing but shifts the question from obtaining the "best" possible signal to extracting the most meaningful information rapidly when the volume of data overwhelms the available computational resources [3].

When discussing large scale data collection, two commonly cited examples of future applications where data is recorded from multiple rapidly sampled sensors are human body monitoring [4] and autonomous transportation systems [5]. In monitoring the human body, the goal is to map and understand the biological functioning of the human body[1] thereby shaping treatments and reducing the discomfort of aging or even extending the lifespan of the human population. In creating an autonomous transportation system, a system that is capable of optimizing the transportation network without human intervention to minimize inefficiencies, we seek to maximize the utility of global resources and efficiently transport goods and services. Both tasks will be facilitated by arrays of sensors that provide continuous feedback and a torrent of information that must be rapidly processed.

The ubiquitousness of sensing technology is changing present day applications, adding detail and reliability when needed. For example, in the domain of vehicle navigation, new safety technologies are demanding greater reliability for the location estimates [6] that GPS cannot provide (ex. driving through a tunnel) [7, 8, 9]. Of the many alternate (supplementary) localization strategies, the best methods invoke sensors whose information content is orthogonal to the GPS signal and is therefore available when the GPS signal is not.

The primary application explored in this thesis is terrain-based vehicle localization. In particular, the motivation for this dissertation is to develop a vehicle localization method using terrain data acquired from a vehicle. Terrain-based vehicle localization can be divided into two phases. In the offline phase, the terrain data is collected by a mapping vehicle and then compressed into a localization map using data representations. Then in the online phase, a second vehicle acquires new terrain data from its sensors and compares this data to the stored localization map. Using this comparison, the vehicle then infers its location.

The localization process in a moving vehicle has three central requirements. First, the process must be computationally efficient such that the vehicle can maximize the size of the map on which it is moving. Second, the on-disk size of the map must be minimized to accommodate the storage requirements in the vehicle. And third, the localization process must be maximally robust to sensor noise such that it can be implemented in production vehicles with cheaper (noisier) sensors.

This thesis takes the approach of representing the data using dynamical models that are of low order, computationally efficient for localization, and tunable to achieve greater robustness to noise. In developing the approach in this thesis, the problem of vehicle localization is extrapolated to the larger data problem of ***in-***

[1] One example is Google's new Baseline Study.

sequence localization. In-sequence localization is the ability to locate a small fragment of a time series inside a larger data set containing the entire history of the process that generates the time series. This is a data-driven problem that is enabled by new data collection methods and that has not been addressed in detail in the literature.

More formally, the problem of in-sequence localization can be described as follows. First, a complete archive containing a data series is collected. This data series is ordered by a flow; typically sequential ordering in time. The data series is complete in the sense that the process repeats and "new" values that are observed follow the historical trends. Then at some later point, a small subset of data from the same process is collected to find its location within the original data set. Here the term location is used to mean the relative index of the starting and ending points of the second subset of data. This problem will be discussed in more mathematical detail in the subsequent chapters.

In-sequence localization is a computationally expensive localization method if the approach used is a brute force comparison of possible subsequences of data [10]. For this reason in this thesis, we assume that during the localization map building the computational resources are unlimited, but during the in-sequence localization, the resources are limited because the process takes place on a mobile platform. Using these assumptions allows the formulations of computationally expensive problems during map building that can be leveraged to reduce the computational cost of the online localization process.

In doing this, the work presented here straddles the previous localization approaches which tended to be probabilistic methods (ex. particle filter) [10, 11] or focused on noise reduction prior to reducing the data dimension [12]. This thesis studies prior dimension-reducing transformations from the literature [13, 14, 15] and evaluates their performance for in-sequence localization. We follow the emerging realization [16, 17, 12, 18, 19] in the data community that data transformations must be robust to account for the noise that is observed in every time series collection process. To this end, the transformations in this thesis are specifically geared to reduce the effect of sensor noise when the distribution of the noise is known. The remainder of this chapter will describe the contributions of this thesis and will provide a road map to the reader of the evolution of the dissertation.

1.1 Overview of Chapters

1.1.1 Chapter 2: Literature Review

Chapter 2 introduces the previous work in vehicle localization and data driven fields such as dimension reduction, pattern detection, fault detection, and filter banks. This introduction provides a context for the remainder of the text. The material presented here covers all chapters, but each chapter hereafter contains an appropriate, self-contained background section of material.

1.1.2 Chapter 3: The Effect of Noise on Alternative Representations of a 1-Dimensional Time Series used for In-Sequence Localization

The problem of in-sequence localization is introduced in Chapter 3. After describing the problem two possible solutions are implemented using seven published and common dimension-reducing data representations. The need for in-sequence localization is motivated by the emerging ability to perform vehicle localization using stored terrain data.

This chapter explains that because in-sequence localization is performed on a mobile platform, the execution of solutions necessitates dimension reduction of the terrain data, noise robustness of the data representations, and rapid convergence to the solution. These criteria are used throughout the chapter to evaluate the feasibility of using the published data representations. The performance of each representation is tested for robustness in translation and for in-sequence localization accuracy. To demonstrate that in-sequence localization is not restricted to vehicle data, the algorithms are also tested on two types of random data. The study shown in this chapter is under preparation to be submitted to IEEE Transactions on Knowledge and Data Engineering.

1.1.3 Chapter 4: A Dynamics Discontinuities-Based Deterministic Data Structure Creation Algorithm for In-Sequence Localization using 1-Dimensional Time Series

Terrain-based vehicle localization using a dynamical model-based data representation is developed in Chapter 4. In this chapter, vehicle localization is performed using only terrain data from a vehicle's on-board sensors. Road data is encoded using linear dynamical models, and then, during travel, the location is identified through continuous comparison of a bank of linear models. The approach presented has several advantages over previous localization methods described in the literature. First, the approach creates computationally efficient linear model map representations of the road data. Second, the use of linear models eliminates the need for metrics during the localization process. Third, the localization algorithm is a computationally efficient approach that can have a bounded localization distance in the absence of noise given certain uniqueness assumptions on the data. Fourth, encoding road data using linear models has the potential to compress the data, while retaining the sensory information. Lastly performing only linear operations on observed noisy data simplifies the creation of noise mitigation algorithms. Preliminary work from this chapter was presented at the 2012 IEEE Conference on Decision and Control [20] and was awarded second place paper at the Penn State College of Engineering Research Symposium in 2012 [21]. The material in the chapter is published in IEEE Transactions on Intelligent Transportation Systems [22].

1.1.4 Chapter 5: Robust AR model-based data structures for In-Sequence Localization using 1-Dimensional Time Series

Continuing the development of dynamical model-based vehicle localization, Chapter 5 introduces one approach to making the representations more robust to sensor noise. The goal is to move noise mitigation from an online process where it requires valuable computing resources to a map building process such that the map automatically reduces the effect of sensor noise.

In particular, in this chapter the dynamical models are fitted to underlying data such that the resulting models are maximally robust to collection sensor noise at the respective model switch points in the data. In essence, the data is optimally divided into features that minimize the effects of noise on the distinction between neighboring model representations. Then, the data representations are organized into structures that further reduce the effects of noise by preserving data order and taking advantage of the properties of uncorrelated noise, namely the fact that averaged data has a reduced standard deviation of noise.

The approach in this chapter is motivated by the application of vehicle localization; however, the development of the chapter pushes further into expanding the developed representation for the more generic problem of in-sequence localization. To consider broader applications, the chapter also demonstrates the applicability of the algorithms by testing them on financial and random data. Preliminary research was presented at the 2013 IEEE Conference on Decision and Control [23] and the 2014 American Control Conference [24] and was awarded the IEEE award for best paper at the Penn State College of Engineering Research Symposium in 2013 [25]. The work presented in this chapter is currently under review in IEEE Transactions on Knowledge and Data Engineering.

1.1.5 Chapter 6: Robust Data Structures for In-Sequence Localization using Multiple Time Series

In Chapter 5 of this dissertation, the robust dynamical model representations of the data are shown to produce smaller localization maps that are easier to compute, and that show that for the given data sensor, large areas of the data do not have a robust model transition point. This suggests that the method by which new sensors should be added to localization maps is to find sensors whose output produces robust transition points in the previously sparse region of the data.

Chapter 6 describes the incorporation of multiple time series when building a localization map. Making a multi-dimensional data representation robust even at transition points is a highly non-convex problem. For this reason, this chapter introduces an approximation that allows the efficient incorporation of multiple time series such that the problem can be solved efficiently as the number of collected terrain data points is increased.

In addition to advancing the theoretical approach to representing multiple time series for in-sequence localization, this chapter also expands the demonstration of possible applications to the approach. Data from two new applications are used for multiple time series in-sequence localization problems: stream health monitoring data, and variable power generation forecasting data. The chapter demonstrates the viability of using in-sequence localization in these applications and demonstrates new possibilities for future research.

The preliminary work supporting this chapter was submitted to the 2014 IEEE Conference on Decision and Control and was awarded the best paper award at the Penn State College of Engineering Research Symposium in 2014 [26]. A manuscript describing the material in this chapter is currently under development for submission to IEEE Transactions on Knowledge and Data Engineering.

1.1.6 Chapter 7: Conclusions and Future Work

We conclude this dissertation by discussing the advantages and disadvantages of our work. We also propose future extensions that would strengthen the approach and add to the body of science.

Chapter 2

Literature Review

Throughout history ideas have been often been discovered and re-discovered only to be forgotten again. In hopes of not falling prey to this scientific folly, this chapter overviews a wide array of previously published papers that pertain to the research in this dissertation. Then each chapter hereafter reminds the reader of the previous research by including an appropriate background section. Chapter 2 is subdivided into three main sections that represent work from four broad categories of researchers: data mining, pattern matching, vehicle localization and histogram filters.

The layout of the data mining section (section 2.1) follows the logical steps taken by a subsequence matching algorithm, the closest and most popular application to in-sequence localization. The first step in any subsequence localization algorithm is segmenting the data in disjoint segments. These segments are then represented by a dimension reducing feature whose purpose is to reduce the computational complexity of matching. Once the data is segmented and represented, new data can be matched or classified using the same dimension reduction technique and a choice of similarity measure. The section concludes by discussing the current directions in the field of data mining. The review here is focused on time domain approaches because this dissertation is developed in the time domain. For an even larger review of data mining papers, some of which are not in the time domain, please see the review article by Fu [15].

Linear model based localization straddles the areas of data mining and pattern recognition. The latter heavily borrows from the first in terms of creating pattern databases and determining similarities in patterns. Section 2.2 in this chapter reviews pattern matching literature that uses linear filter banks for detection. In particular, the papers review are focused in texture detection which is the main area of patter matching that uses linear filter banks.

Because the main application for the dynamical model-based in-sequence localization technique described in this thesis is vehicle localization, section 2.3 reviews relevant work in localization. This work can be thought of as pattern matching in vehicle data, where the relative uniqueness of the patterns determines the quality and the speed of the localization process. This section also reviews previous work by other researchers in the authors' labs. The chapter concludes with a brief description of histogram filters in section 2.4.2, which can be thought of analogues to the localization process developed in this thesis.

2.1 Data Mining Background

2.1.1 Segmentation

In general terrain-based localization or the more general problem of in-sequence localization are not computationally tractable using a burte force subsequence matching approach. For this reason the first step in terrain-based localization (in-sequence localization) is the creation of a reference map that contains patterns which represent salient information in the data. The data mining community has frequently used linear regressions in the past to models trends and patterns in the data. In particular the work of Shatkay and Zdonik [27] showed that linear regressions can be used to reduce the data dimension to represent large data sequences. The work by Shatkay and Zdonik aims at determining an approximate match to a subsequence using the linear regressions and then verifying the match using the complete subsequence which is stored in parallel with the representation. To keep the matching computationally simple, on the first order regression is saved as a representation. The use of larger order models is rejected as too complicated, and generally less efficient. Simplifying the representation of data even further, this paper further suggests the representation of data using only the extrema points. However, the authors determine that using only extrema points is too simplistic to fully represent the data.

Following the approach in [27], many authors have used variants of the first order regressions. Prominent examples include the piecewise aggregate approximation (PAA) [28, 29] which uses the mean values of a series of data segments as representation. As noted in the literature the main drawback of the PAA algorithm is its fixed segmentation length that may intersect important features in the data. This limitation is subsequently addressed in the adaptive piecewise constant approximation algorithm (APCA) [30]. A second representation that adaptively models the data is the piecewise linear representation which is a set of slopes that describe data segments chosen to minimize the mean square error between the representation and the data [31, 32]. Specific descriptions of these algorithm and others to follow can be found in Chapter 3 where the techniques are implemented for testing for in-sequence localization.

In general, segmentation is the determination of either the extrema of the time series or the "significant" points that enable pattern detection. The term significant is defined in each publication by the authors. Significant points were first defined in [33] as the points at which the behavior of the time series changes. The problem is analogous to that called "change point detection" in the statistics community. The generic approach to solving this problem is to identify a set of break points through the identification of a set of models about the point. These models are identified by minimizing a loss function specified by the author. No single loss function has emerged as the correct choice. Popular choices to be described below include mean square error, least squares fit, and maximum likelihood estimation. The problem is most complex when encountering situations which require streaming segmentation of segmentation where the length of the extracted segments is not identical. These latter cases are the currents trends in development of the field.

Once a significant point (or transition point as will be later discussed in this thesis) is chosen, the data is bisected about the point during the segmentation process [34]. It is notable to mention that the idea to bisect significant points was proposed earlier in [35]. In this paper the switching sequence of subprocesses was identified using nonlinear gated experts, a statistical physics tool.

Concurrently with the development of the approaches above, a dynamical programming approach [36] was developed to find the number of possible data intervals, the model order in each interval, and the location of the intervals. This work was aimed at determining the best possible method of modeling a set of data. Duncan *et al.*'s algorithm [36] is based on a least-squares fit of the models onto the available data. It is a more efficient method to detect of changes in dynamical systems than the one proposed in [37] which is based on the parameterization of linear systems in [38].

There are also several notable sliding window approaches that compete with the significant event detection algorithms described above. First, [39] presented a sliding window approach to change point localization and segmentation. Then, in [40] piecewise segmentation and identification is proposed. The approach maps similar segments in a time series as neighbors in a neighborhood map. Then later, in [41] a bottom up sliding window method termed SWAB is suggested.

2.1.2 Robust Features

The literature on robust data representations, or data representations that are designed to counteract the effects of noise, is sparse. Few papers explicitly discuss or make mention of data noise as a consideration. One possible reason for this lack of literature is that most data mining is performed with the goal of finding "similar," not exact features. Another possible reason is that the signals on which these algorithms are based are strong, with large signal-to-noise ratios (SNRs) that minimize the effect of noise. Both of these assertions are not valid in the application of vehicle localization where an exact identification is needed and where the terrain data has relatively poor SNR when the decimation of the recorded terrain data is on the sub-meter scale.

The most commonly discussed representation where noise was observed in the data is the piecewise linear representation (PLR) [42, 43]. In particular the work by Jia et. al. [43] contains a good review of PLR algorithms and then aims to develop a new approach that improves PLR's robustness to noise. Here the authors note that the need for a user-specified number of segments is a major weakness. By reformulating the problem in terms of error bounds, the authors automatically choose the number of segments in PLR on a data set given an error bound. The advantage of formulated the error in terms of error bounds also extends to noise robustness because the error bound could be chosen such that the level of noise is accommodated.

A separate subset of papers by Fink et. al. [44, 45] also focuses on the problem of identifying noise robust points. The difference here is that the papers focus on the identification of maxima and minima in the data, which are inherently more robust. The paper by Fink and Pratt [44] builds on this idea by identifying patterns in the reduced dimension (extrema point) data set.

Building on the idea in [44], the work by Vemulapalli et. al. [12] proposes an optimal filter that reduces the effect of noise prior to identifying patterns. The patterns in this work are combinations of robust minima and maxima in the data. Combining filtering and robust pattern identification improves on the classical approach of smoothing the data before obtaining representations. Furthermore, this approach is verified using real-world vehicle data from the author's lab.

Lastly, a more general approach to robust data representations laid out by Preng et. al. [46] cites the natural method of location recognition by animals and humans, which use landmarks to identify particular sites in their surroundings. The landmark model, as it is called in the paper, uses landmarks as robust data points that can be used for identification. Specific development for a specific application is left to future authors.

However, recognition of both the problems introduced by noise and the limitations of sensor technology is increasing. Application specific papers from several domains offer insight into this problem and evaluate the performance of algorithms specifically evaluated with respect to the quality of sensors. In the vehicle localization domain the authors of Stoyanov *et al.* [16], Mullane *et al.* [17], and Vemulapalli *et al.* [12] discussed the effect of sensor noise on the building of localization maps. In the realm of air traffic control where the piecewise linear representation (PLR) is used to model aircraft motion data, Guerrero *et al.* [18] discusses the limitations of PLR with respect to noise and and suggest improvements to the PLR algorithm handle noise. Lastly, dealing with optical data, Skauli [19] discusses the effect of sensor noise in storing and processing hyperspectral data that is recorded from optical sensors.

2.1.3 Subsequence matching

When segmenting a time series the resulting representation is often used for subsequence matching, or as discussed later in this thesis for in-sequence localization. Subsequence matching is the identification of a user specified subsequence from existing data. And in-sequence localization is the matching of the subsequences and the identification of the time series point locations in the previously recorded time series. This section will describe the existing methods of subsequence matching.

Initial work in the field of subsequence matching was focused on the matching of whole sequences [47]. This work is focused on reducing the dimension of the data by taking the discrete Fourier transform and preserving the first 2-3 coefficients. Using Parseval's theorem the authors note that Euclidean distance in the frequency domain is the same as the Euclidean distance in the time domain. They then use a Euclidean distance in the Fourier coefficients to determine a match between the query sequence and the database sequences.

Building on this work, the authors of [48] generalize the algorithm to subsequences, i.e. fragments of time series that are smaller than the originally recorded sequence (time series). The developed algorithm, often referred to as FRM after the author's initials, uses a sliding window approach to find all possible subsequences in the data. A sliding window approach can be thought of as a fixed width window, that is slid across the time series one end point at a time. Each new offset point generates a new subsequence and a new set of Fourier coefficients. Because the coefficients are similar at nearby offsets, groups of coefficients are collected in minimum bounding rectangles (MBRs) that are stored hierarchically in a tree to enable quick localization.

Together the papers [47, 48] form the basis of the data dimension reduction research community. Since these papers, many authors have sought to improve the performance of subsequence matching. An interesting example of this is the work in [49] that is the dual of the FRM solution. Here the algorithm called Dual Match segments the database into disjoint windows and the query sequence. This segmentation is performed using a sliding window. The idea behind this algorithm is to use the Fourier coefficients directly while storing the tree structure, instead of the MBRs. This eliminates coefficient uncertainty created by MBRs and reduces the false alarm rate.

Reducing the false alarm rate is a critical component of increasing algorithm speed and efficiency, because the largest computational cost comes from the overhead generated by the post-processing required to eliminate false alarms [50]. There exist several papers that transform the storage structure to optimize CPU usage and minimize bottlenecks [51, 52].

There are also other approaches in the literature that perform subsequence matching based on the Euclidean metric but are not based on the work in [47, 48]. For example in [53] the authors propose an approach that entails modeling the query and database data using slopes that connect minima and maxima in the data. The query is also described using a similar set of sloped lines. Subsequence matching is then performed by sliding the query sequence representation over the database representation at all offset points, and comparing the Euclidean metric at each instance.

Another example is the work in [54] which proposes to extract anomalies in time series data which can be used in classification and matching. This work focuses on the creation of an efficient algorithm to detect these anomalies because the original brute force problem is computationally prohibitive. The references here in also point to a different body of work, which seeks to discover patterns or motifs in time series data.

Lastly, there are also subsequence matching algorithms based on PLR [55, 56]. A key feature of these algorihtms is that they are online algorithm, i.e. algorithms that are capable of processing data in real time. Testing their performance, the authors use streaming financial data. Incoming data is modeled using a linear piecewise representation, and then matched to the query sequence using a permutation of the each, the set of higher segment bounding points, and the set of lower segment bounding points. It is notable here that the metric is similar to other subsequence matching literature (e.g. the Euclidean metric).

In fact, the majority of work cited in this chapter to date has relied predominantly on the Euclidean metric. This is because the Euclidean metric is a relatively computationally inexpensive approach as compared to metrics used in the field (e.g. dynamic time warping (DTW)). Using DTW, a method that aligns time series on the time axis, several authors have presented notable results [57, 58, 59]. We choose not to focus on this work because we are interested in developing approaches that rely on the smallest number of pre-processing steps and the lowest computational cost.

Lastly, this section would not be complete without a short overview of the benchmarking papers that evaluate the state of research in the data mining community. In [13], Keogh and Kasetty offer a scathing review of the data mining community and the approaches presented to data. Of 360 initially reviewed papers, only 57 are cited in the final study. Of this, the majority perform only marginal comparisons to other work and exhibit strong biases such as implementation bias and data bias. Furthermore, it is interesting to find that at least 70% are based on the initial work in [47, 48]. The paper concludes with some suggestions to improve the quality of papers published by the data mining community.

An updated comparison is co-authored by Keogh in 2008 [14]. This paper argues strongly for the adoption of tightness of lower bound metric by which to evaluate time series representation. This metric bounds the minimum Euclidean distance between extracted time series features, ie. if we imagine the features as balls in a two dimensional space, then the minimum bounding distance is the smallest distance between two balls. The conclusion drawn by the authors is that this minimum distance is strongly correlated to the ultimate number of subsequences retrieved from storage, with closer subsequences requiring more queries and *vice versa*. In addition the authors evaluate the full breadth of the similarity measures proposed in the literature. They conclude that as the size of the data set grows, all similarity measures converge to the Euclidean metric performance. Due to the relative simplicity of this metric, there is then no advantage to using more complex approaches suggested in the literature.

2.1.4 Data Compression via Pattern (Feature) Extraction

2.1.4.1 Rapid On-Line Clustering and Discovery

One of the big challenges in the data mining community is the online compression and classification of streaming data time series. This need is driven by the increasing availability of data from a growing array of sensors. As the data increases, the storage requirements also rise rapidly and the pattern discovery process grows exponentially in computational cost [15].

The first online algorithm presented here is by Fu et.al [60], which introduces perceptually important point representations algorithm for stock data. In this paper the authors argue that the data representations rely mostly on a few extrema points and therefore the in between fluctuations can be safely discarded and approximated by line segments. The resulting features are then organized using self-organizing maps (SOMs) that cluster the extracted patterns according to a distance (squared Euclidean distance) from pre-specified set of nodes. Following the addition of a pattern to a node, the node itself is updated hence the SOM term.

The approach of neural clustering of SOMs was first pioneered by Kohonen [61, 62]. In terms of clustering, SOMs have the desirable property of emergence which is the ability to discover new patterns that have not been pre-specified by an expert [63]. This property of the algorithms follows from the update step of the network.

Other researchers have also adopted SOMs, for example Euliano *et al.* [64] describes one approach to SOMs in the presence of wave distortion in time and space. In the medical field [65, 66] use the emergence property of SOMs. The first paper discusses the effects of the discretization on the identification of persistent states in the SOM. A persistence score is introduced to aid in the author's argument. In the second paper the authors use SOMs to visualize the data better for human comprehension. Returning to more general data mining, [67] proposes to replace the raw data with measures such as trends, seasonality, serial correction, etc. and then to use an SOM to classify the data. In the financial domain SOMs are employed in [68, 69] to determine stock closing price patterns.

2.1.4.2 Pattern Analysis Using Linear Models

Linear models have also been used extensively in pattern extraction from time series. The papers listed here and the references therein describe the linear modeling approaches taken in the past. In particular [70] employs the concept of the linear prediction coding cepstrum. The linear prediction coding cepstrum is the cepstrum of a model derived from the coefficients of an auto-regressive model. The authors of this paper use the cepstrum to define a distance between models and then use this distance for clustering. In [71] linear models are used to demonstrate that clipped data approaches the same modeling coefficients as unclipped data. This helps improve the efficiency of clustering algorithms by removing outliers and replacing them with rounded values.

Linear models are also appropriate for variable length patterns. The authors of [72] use ARMA models and an expected maximization algorithm to determine models that appropriately represent time series data. Other authors have also focused on the variable feature size of patterns. Examples of this that are not linear models include [73, 74]

2.1.5 Classification

Time series classification is a traditional task handled by the data mining community. This task appears as a subtask above in the segmentation and subsequence detection work. For example, Geurts *et al.* [75] model the time series using piecewise linear models and extracts the break points between line segments as temporal features for classification. In doing so, this algorithm both reduces the size of the data and retains perceptually important points for classification. The latter is key because of the stated goal of the author to maintain features that are perceptible to humans.

A popular approach to classification is through the extraction of features using wavelets. In [76], the authors extract features from wavelet coefficients. As pointed by the authors wavelets preserve the Euclidean space between time series in the time domain. The distance between wavelet coefficients is then generalized to a distance between time series of varying length.

Other approaches include classification by modeling [77]. In [77] the time series data is normalized and classified according to its reconstructed phase space. Reconstruction of phase spaces is a method of indirectly reconstructing the phase spaces of a dynamical system. The phase space itself contains all system states. Once the data phase space is reconstructed, Gaussian Mixture Models (GMMs) are used to fully describe all aspects of the space. Then during classification a Bayesian maximum classifier is used to detect test signal class.

Classification is typically performed by first extracting a data structure that is later used to compare test series. Rodriguez *et al.* [78] argue that using raw data in data trees leads to poor results. They present two approaches, one that creates an interval tree with each interval described as data means, and a second approach creates tree after first dynamically time warping the data. The authors find these approaches to be competitive with previous work. Lastly, many authors have also focused on the time series data set size reduction which is necessary to make all of these methods feasible for implementation [79, 80].

2.1.6 Current Directions

The field of time series analysis has evolved dramatically in the past decade. Prior to the early 2000s the field was concerned with time series segmentation and matching. Strong arguments were made for metrics and segmentation algorithms that were eventually consolidated two review papers [13, 14]. In summary, representations with complexity greater than first order were deemed too complex, and all measures for subsequence matching converge in performance to the Euclidean metric.

In the 2000s the field moved on to exploring new approaches such as the one nearest neighbor approach (1-NN) and self organizing maps (SOMs). Lately, one can observe that the field is moving towards both time series of greater length, series that are analyzed on-line (or streaming series) and multiple dimensions. The final chapter of this dissertation also moves in this direction. The work presented in this chapter focuses specifically on the incorporation of multiple time series in data representation. For this reason this section focuses on previously published research in multidimensional time series.

There are two types of multidimensional series: time series with multiple attributes and truly multiple time series are may be correlated. The difference between multiple attribute time series and multiple time series is almost a semantic difference. For example, in the world of finance data, a stock's volume and price data can be considered two time series, or a single time series with multiple attributes. The latter view was taken by Povinelli and Feng [81]. In this work the authors choose to combine the attributes of stock volume and price into a phase space. Data clusters are identified in the phase space by determining the optimal data means via a linear optimization problem. In essence, clusters of data with sufficiently dissimilar means are identified to denote evens in the time series and non-events. The goal of the work is to address previous deficiencies in data mining algorithms that assumed stationarity in the underlying data.

A similar approach is taken by Kahveci *et al.* [82]. In this work the authors focus on the development of a shift and scale invariant clustering and indexing of the multi-attribute time series. The authors begin by showing that the Euclidean space of time series is often not robust to shifts and scaling. The subsequences are then clustered by similar shift and scaling properties. In multiple dimensions this appears as a cone with clusters appearing as slices of the cone.

Focusing less on invariance and more on the pattern uniqueness, Lee *et al.* [83] creates a data mining approach that k dimensional patterns from m dimensional data. The basic idea is to find 1-dimensional patterns and to concatenate them to find patterns the smallest unique frequent patterns in a multi-dimensional database. It is important to note that this work is applicable to both multi-attribute and multi-dimensional time series as the patterns are generally the same under the author specifications.

Moving more towards truly multi-dimensional time series Minnen *et al.* [84] address the interesting problem of data patterns that not fully dimensional. By this we mean that in an m dimensional data set, the patterns have a dimension less than m. The paper by Minnen *et al.* focuses on the efficient discovery of these patterns.

Simultaneous work by Minnen *et al.* [85] also extended the use of Hidden Markov Models (HMM) into the multidimensional motif discovery. In a scalable algorithm, ie. one applying to single and multiple dimensions. The algorithm begins by extracting all subsequences of a fixed length. Then a preset number of nearest neighbors is determined for each sequences. Effectively this clusters the subsequences, in order to determine their estimated densities. Finally an HMM is trained using the clustered subsequences and their respective densities. The states of the HMM represent the motifs that have been discovered in the data.

The inherent clustering of subsequences in [85] is also a major theme in multi-dimensional time series analysis. For example the work by Plant *et al.* [86] focuses on the clustering of multidimensional time series based on the interaction between the dimensions. This is an interesting approach because it preserves perhaps the most important information from multiple dimensions. To accomplish this, each dimension is modeled from other dimensions using linear models. The clusters are then time series data that exhibits similar interactions between the dimensions.

Similar to the work in [85], Tatavarty *et al.* [87] propose clustering time series data based on the temporal similarities between frequency patterns. To accomplish this, the authors first identify and cluster all frequency patterns in each dimension of the data. The dependence among patterns in different dimension is then identified. The goal of these two works is to extract the greatest amount of information from the data.

Another interesting clustering algorithm is presented by Wang *et al.* [88]. In this algorithm motion time series is clustered based on spectral features. These features are found to be more robust normal spectral clustering. Spectral clustering in general is a method of reducing the data dimensionality using the eigenvalues of the similarity matrix before clustering itself.

Yet another interesting application of multidimensional time series analysis is prediction [89]. The paper by Shibuya *et al.* [89] and the references within, demonstrate the use of multidimensional time series to predict behavior. A key component in this research is determining the causality of one dimension on another. In other words, researchers are looking to determine whether the fluctuation in one time series can be used to predict the behavior of another.

The research directions of similarity searching [90] and tree structuring [91] are also addressed. These areas are important and still open areas for research that bridge the divide between theory and practice. Finally, in terms of vehicle technology, there are some examples of work that incorporate multidimensional databases. The work by Gonzalez *et al.* [92] is one such example where multidimensional traffic data is mine for anomaly detection. These anomalies are used by traffic engineers to detect accidents early and re-route traffic to maintain flow along the nation's highways.

2.2 Pattern Matching/Texture Detection

As discussed in the introduction of this chapter, a second particularly relevant area of research to this thesis is pattern matching. In parallel with the data mining community which has used linear models for subsequence matching, the pattern matching community has also used linear models for pattern detection in applications such as texture classification and tracking [93, 94, 95, 96, 97, 98], shape detection [99], fault detection [100], acoustic filtering [101, 102, 103], and finger print identification [104].

The two types of texture detection that are most relevant to this thesis are dynamic texture detection and single frame texture detection using linear dynamical models [105, 106, 107, 108, 109, 110]. Dynamic texture detection is the ability to detect a texture evolving across several frames, while single frame texture detection looks at identifying a texture within a single frame using dynamical models. Using dynamical models in these applications is advantageous because many evolving textures exhibit linear dynamical behavior and the use of these models provides access to well developed tools from the control system literature [95].

The general approach when using linear dynamical systems for texture detection is to extract a set of linear models from training data that represent the desired textures. Some authors employ switched linear systems to obtain this reference set [109, 111]. Then using new data, new linear models are extracted and then matched to the reference set to determine the exact texture. The metric used in matching is often in the space of linear dynamical system coefficients such that the distance between the model coefficients is interpreted as the distance between the textures [110]. An elegant metric that is often used in this domain of research is the Martin metric [112] that measures model distance in the cepstral domain, which as previously discussed is additive and a function of the poles and zeros of the system. Other authors use control system theory concepts to develop robust model validation techniques [95], and still others use optimization techniques to determine the model feasibility in the presence of uncertainty [107].

In this thesis the vehicle localization procedure can be thought of as the sequential matching of patterns which in our case are patterns of terrain pitch data. This differs from approaches taken in the texture detection community in several important ways. First, texture detection approaches are static in the sense that they assume the data is available before the detection process. Second, the presented approach does not require the use of matching metrics which are pivotal in texture detection algorithms. Lastly, to identify the correct vehicle location, our approach requires the correct identification of several sequential linear models. Thus we identify a series of textures rather than a single specific texture. Previously literature has shown that identifying a series of models increases the certainty of the algorithm output. In addition, as pointed out in [94] using a sequences of linear systems also reduces computational complexity by allowing a reduction in model order.

2.3 Pattern Matching in the Vehicle Community/Localization

This chapter will now conclude with a brief overview of vehicle localization research that will show the reader the contributions made by this thesis.

2.3.1 Sensor-Based Localization

Sensor based localization is the localization of a vehicle (robot) using only the sensory data from the platform. There are two types of localization: reference localization (or dead-reckoning), where the vehicle location is propagated from a known position, or relative localization, where the vehicle position is inferred from environmental landmarks.

The three sensors commonly used for dead-reckoning (DR) localization are the odometer, the velocity encoder and the inertial measurement unit (IMU). A series of review papers can be found in [113, 114, 6]. Perhaps the simplest method of navigation is odometry. To keep track of odometry, manufacturers install a sensor that records a pulse each time it passes by a specific point on the wheel. The number of pulses is then multiplied by a scale factor that is related to the wheel diameter, tire pressure, temperature, and any environmental factor that changes the wheel radius. References citing the use of odometry and error-correcting techniques in vehicle localization can be found in Abbott and Powell [114] and also within the mobile robotics community [115, 116].

A similar set of sensors are the wheel tachometers. These sensors are typically employed in anti-lock brake systems (ABS) to detect the differences in wheel speeds. Using the same framework, the difference in wheel speeds can reveal the direction of the vehicle. Like the odometer, the critical piece of knowledge necessary to make use of velocity encoders is wheel radius. In Carlson et.al. [117, 118] velocity encoders are used in a dual GPS/DR system. Here while GPS is available, the signal is used to provide a heading and a wheel radius estimate. When GPS becomes unavailable, wheel tachometers are calibrated to provide heading information.

Odometers and velocity encoders suffer from well-known sources of terrain error such as wheel slips, uneven road surfaces, and skidding. In addition, these same sources suffer from vehicle structure errors such as wheel diameter changes, wheelbase uncertainty, and low resolution of the encoding sensors. These errors have been observed and characterized both in the vehicle navigation community [114, 118] and the mobile robotics community in Borestein and Liqiang [115].

In contrast to odometry and tachometers, gyroscopes and accelerometers in IMUs measure rates of acceleration and rates of change in displacement of the vehicle. A typical IMU consists of three gyroscopes and three IMUs, each directed in one of the axes of motion. To obtain vehicle heading and velocity, the IMU outputs are integrated in each direction. This integration leads to the cumulative growth of small bias and noise sources in the IMU as a function of the operation time. These noise sources will be further discussed in section 4.2.1.2. However, because IMUs are self-contained and thereby not susceptible to the vehicular and environmental noise sources, IMUs are a good complement to other vehicle sensors and GPS [119, 120]. For this reason, GPS and IMU data is frequently integrated in augmented systems [121, 6].

2.3.1.1 Map-Matching

The DR localization approaches described above can be used to estimate vehicle location given a known starting point for the vehicle. However, if the location is unknown, then the vehicle position can be estimated through the comparison of sensory data with a localization map. This map can be of beacon locations, requiring triangulation for localization (the GPS is a form of active beacon), or of another type of feature which is selected by the localization mechanism designer [122, 123, 124]. This form of relative localization is called map-matching.

In this thesis, localization maps are built using linear models and the localization mechanism combines DR using inertial measurements and map-matching. A state of the art review of map-matching is found in Quddus et.al. [124]. The author subdivides the field of map-matching into early techniques that take into account the road geometry, such as arcs and lines; topological approaches which take into account road geometry and the interconnection between each geometric feature; probabilistic approaches that assign a region or error within which a likely road segment is found; and more recent approaches termed advanced map-matching using Kalman Filters, fuzzy logic, particle filters, etc. The references therein provide an in-depth look of the field of map-matching.

As a general outline, the map-matching process begins by extracting or observing features from the data [125]. The choice of features during the extraction process has a pivotal role in the subsequent localization performance of the algorithm. Early algorithms use arcs and lines as features that modeled the road shape [122, 126]. In advanced algorithms, the features are extracted through a non-linear transform that is used to identify the most noise-robust portions of the data. A few examples are local extrema, geometric features such as arcs and lines, or geometric beacons. Further examples can be found in the literature on pattern matching and data mining [127, 128, 129, 130, 131].

Then, during localization, the features are re-extracted from new vehicle data. The location is determined by comparing these features to the map. The most common comparison approach is the threshold. Here a distance is computed from the road data to each possible map location. An empirical threshold is used to determine the correct segment. More advanced methods of matching include those presented in the robotics community where the probability of the vehicle's location and of the detected features is calculated using Kalman filtering in a formal framework methodology established in [132, 133, 134], or a Bayesian comparison approach such as the one used by Levinson and Thrun [135], or a particle filtering approach in Törnqvist *et al.* [136].

The motivation to develop advanced map-matching algorithms stems from the need to handle increasingly complex road networks that easily overwhelm earlier algorithms. Unfortunately, to the best of our knowledge, there does not exist a unifying publication that elucidates the limitations of each map-matching approach in the vehicle navigation community. In contrast, several review papers in the data mining community have concluded that more work is necessary to design efficient algorithms to handle large databases [14].

Choosing the features and the extraction process plays a pivotal role in the subsequent performance of the algorithm. Previous research in our own group has focused on IMU pitch data for localization [10, 137, 138, 119, 120, 127]. In particular, Dean [10] demonstrated a particle filter-based approach using inertial measurements. A more noise robust approach was presented in Vemulapalli *et al.* [137] where an optimal filter was derived to process the data prior to extracting extremum features. Combining previous work, the work by Katetotad *et al.* [138] employed both the particle filtering and feature-based approaches to allow localization and tracking of a vehicle over a greater area.

A third alternative is called Simultaneous Localization and Mapping (SLAM). This alternative is the simultaneous building of the localization map and the concurrent vehicle localization [139, 140]. The SLAM approach is mostly advanced by the robotics community.

2.3.1.2 Noise Characterization of the Inertial Measurement Units

The main sensor used in this dissertation is the inertial measurement unit with readings such as vehicle pitch and roll playing pivotal role in the creation of the data representations. For this reason this section introduces recent work describing the use of IMUs in localization and the characterization of IMU noise.

Recent decreases in the cost of Micro-Electrical-Mechanical Systems (MEMS) IMUs have contributed to a rise in the number of publications citing augmented GPS/IMU systems [141, 119, 120, 142]. Given this development and the potential for further research in the future, it is important to understand how sensor limitations affect the localization approach. In particular, El-Sheimy *et al.* in [143] characterize IMU noise in terms of Allan variance and power spectral densities. The same authors then offer approaches to mitigating IMU noise [144]. A MEMS specific analysis is demonstrated by Aydemir and Saranli [145], while the work in [146] evaluated the noise performance of several grades of IMUs.

Extensive research has also been presented on the calibration of IMUs [147, 148, 149]. An in-depth discussion of IMU noise components is shown in section 4.4.3. This description is then used to corrupt the IMU data for localization simulations.

2.4 Additional Topics

We conclude this section by discussing two more topics seen in the literature: fault detection and histogram filters. This work is related to this thesis but not as strongly as the work above. For this reason the background here is abridged to briefly introduce the reader to these topics.

2.4.1 Fault Detection

In contrast to the data mining community and the pattern detection community, this task is significantly more difficult. The reason is that the field of fault detection is fragmented into application subfields. Few papers have ventured to unite the subfields into coherent general bodies of work. Examples include [150, 151, 152, 153, 154].

In general the majority of the fault detection work originated with plant models of designed systems. These plant models had available inputs and output and naturally led to the adoption of control system techniques for stabilization and monitoring. These techniques include state and output observers which observe the states of the known system and detect changes that indicate faults. Also, parity equations which is a way to compare a system's input/output behavior to the expected transfer function [155]. Lastly, parameter estimation is the continuous re-estimation of system parameters until a significant deviation demonstrates a fault has occurred.

A subset of model based fault detection is signal-model-based methods. This subset of methods applies in the case where only plant outputs can be observed. In this case researchers focus on filter based methods like spectral analysis, bandpass filters, and maximum entropy estimation to identify faults that produce novel frequency components. Yet another subset of techniques are called the change detection methods. These methods are largely probabilistically based and detect changes in the probability mean and variance of the plant output, or use statistical techniques such as a likelihood-ratio-test or Bayes decisions.

2.4.2 Histogram Filters

The algorithms developed in this thesis continuously validate the road pitch models with the incoming vehicle data. Each feasible model is considered feasible until contradictory data is observed by the vehicle's sensors. This can be viewed as a binary "on/off" decision, and correspondingly path-wise as a binary histogram filter.

Histogram filters are numerical implementations of the Bayes filter over an area that is discretized into some appropriate number of bins [156]. The key advantage of discretizing the search area is that this allows the derivation of recursive equations to implement the discrete Bayes filter. The recursive implementation is computationally efficient for online operation. However, there are two underlying assumptions that must hold when implementing the histogram filter. First, all possible predictions from the data can be summarized by the most recent data point. And second, the data and predictions must obey the Markov property.

Histogram filters are appropriate for vehicle tracking and localization due to several properties of the vehicle system. The most significant property is order which is enforced by the physical nature of the process. In essence, vehicle states such as velocity and position always evolve sequentially from previous states.

Several studies have presented compelling evidence that approaches similar to the histogram filter are appropriate for vehicle localization and tracking [157, 158, 159, 160]. In [157] an algorithm is developed for the localization and tracking of a land vehicle from an unmanned air vehicle (UAV). This algorithm uses a directed graph and breaks up the vehicle location into discretized states that are updated using a Bayesian filter. The algorithm builds on previous work [158] that implemented Bayesian filters to localize vehicles with a known velocity and final destination within a mine. Bayesian filters were also implemented in [159] to track vehicles from multiple UAVs. Additional tracking work was presented in [160] where a histogram filter tracked underwater targets using sonar.

The work of these authors demonstrates that the histogram filter is an appropriate tool for vehicle localization and tracking. While this thesis does not implement a histogram filter, the analog between the work here and histogram filters is strong because, along a path, each model agreement can be considered a "vote" for the current segment. Unlike histogram filters, in this work votes are not presently aggregated over all models. However, future implementations of the presented algorithms may incorporate a histogram filter as a recursive method of estimating the correct vehicle path from among several surviving possibilities.

Chapter 3

The Effect of Noise on Alternative Representations of a 1-Dimensional Time Series used for In-Sequence Localization

The goal of this chapter is to formally introduce the problem of in-sequence localization by collecting seven published dimension reducing data representations [13, 14, 15] and evaluating their performance in the problem of in-sequence localization: the Piecewise Aggregate Approximation [28, 29], the Discrete Wavelet Transform Representation [161], the Symbolic Aggregate Approximation [54], the Discrete Fourier Transform [47], the Chebyshev Polynomial Representation [162], the Piecewise Linear Representation [56], and the Adaptive Piecewise Constant Approximation [163]. These representations, and the procedures to obtain them from a data set, are described in some detail in the text, creating a survey of available dimension reducing representations that may be applicable for in-sequence localization.

The evaluation of the performance of a given representation for in-sequence localization begins with evaluating the translation of the representation before and after the addition of noise. Testing is performed on four different data sets: two vehicle data sets and two synthetic data sets. Then, an in-sequence localization procedure is demonstrated for each representation. Two general types of procedures are shown, a fixed window size procedure and variable window size procedure. In each case the performance of the representations is tested across all four of the test sets discussed above. Lastly, the chapter concludes by suggesting two possible approaches to creating dimension reducing representations for in-sequence localization. The first approach is to incorporate generic noise descriptions that fit the application [20, 23, 24, 22] and the second approach is to incorporate process specific information about the application [18].

A manuscript detailing the work in this chapter is currently under preparation for submission to IEEE Transactions on Knowledge and Data Engineering.

3.1 Introduction

The work presented in this dissertation is one possible approach to addressing a pivotal challenge of modern data processing: the determination of formal methods to reduce the dimensions of data to a meaningful set of features (representations) that are then stored in logical and efficient data structures. While previous research in dimension reducing data representations exists [28, 29, 161, 54, 47, 162, 56, 163], there are two principal challenges that face the data community. First, the existing methods have been repeatedly shown to scale poorly with time series size [13, 14]. Second, the unprecedented scale on which data is presently collected is enabling new applications that demand a fundamental rethinking of dimension-reducing representations.

But these challenges in data dimension reduction are not independent. The newly introduced problems are challenges that help describe theoretical deficiencies that may have previously been obscured. This chapter formally introduces one such illuminating problem, the problem of in-sequence localization. Our interest in this problem is motivated by the problem of vehicle localization using large data. This intuitive application is revisited throughout the thesis to illustrate ideas and concepts.

Localizing vehicle using in-sequence localization is a difficult problem because of the extraordinary size of available data. For example, presently an autonomous vehicle may collect up to 750 Gb of data for each second of driving[1]. Because the average road vehicle is driven 600 hours per year, this means that an autonomous car would collect 1.62 Exabytes (EB $= 1000^6$) of data annually, and all this data must be reduced and stored such that it enables in-sequence localization.

In addition to size, noise introduced in the data via the collection sensors is also a significant problem, [16, 17, 12, 18, 19]. This noise obscures trends in the data and makes in-sequence localization using terrain data particularly challenging. In general noise increases the computational cost of localization because it requires the addition of mitigation algorithms that improve the signal-to-noise ratio (SNR) of the data. Thus the problem of in-sequence localization is the problem of reducing and representing large data sets such that the resulting data structure is not obscured by noise.

The solution to the problem of in-sequence localization is divided into two phases - an offline phase where computational power is unconstrained and an online phase where computation is constrained and the localization is taking place. In the first phase a complete time series data set is collected. This data set describes a desired process in its entirety, for example in vehicle localization this data set is all terrain data from a road or road network. The time series data is then represented using a dimension reducing representation and organized into a data structure (ex. significant road features are organized in a map). In the second phase, a mobile platform is localized using the created database (ex. finding the location of a vehicle on the map). Here newly acquired data is matched to the stored data structure, and successive data points are matched to the previously feasible points in the map. The successive matching helps to narrow the location estimate to the correct location. In addition the approach of successive matching reduces the computational requirements on localization.

Localizing only the most recently-acquired data points implies using knowledge about the data ordering [20, 23, 24, 22] that is not typically used in dimension-reducing data representations. In fact, when evaluating typical dimension reducing representations [28, 29, 161, 54, 47, 162, 56, 163] for in-sequence localization, three criteria emerge from the flaws of the previously published data representations.

[1]This is the amount of data estimated to have been collected by the Google autonomous vehicles during the XPrize competition in 2013.

First, the representations must allow the initial data samples to be located in the middle of a data interval. Second, the data representations must preserve the ordering (flow) of the data. Third, the data representation must have an inherent robustness to noise. The solutions to the problem of in-sequence localization that meet these criteria can be applied to other data rich areas such as the identification of system states, estimation of historical trends, or retrieval of data from a collection [50, 164, 165].

3.1.1 In-Sequence Localization in Vehicle Data

The problem of in-sequence localization in vehicle data is that of finding the vehicle's location in a previously recorded data set. This application is commonly called vehicle localization, and it is more precisely defined in this section. The text here serves as an overview for the vehicle data simulations within the remainder of the thesis.

Consider the terrain data-collection process in a vehicle that is illustrated in Fig. 3.1. In tandem with this illustration, Fig. 3.2 shows a block diagram representation of the mapping of vehicle position into pitch. In these depictions, the variable $v(t)$ represents the vehicle velocity; $s(t)$ is the displacement of the vehicle; $p(t)$ is the position of the vehicle, relative to its initial position $p(t_o)$; and $m(t)$ represents vehicle pitch that is mapped from the vehicle's position. The terrain data collected by the vehicle is represented by $d(t)$ and it is subject to additive sensor noise $\eta(t)$.

During localization, the noisy data is then matched to a stored database. More precisely, for each instant t the data point $d(t) + \eta(t)$ is compared to the map and possible matches are identified. At first, the data point is compared to the whole map, as parts of the map are eliminated, comparisons only take place on the remaining possible regions. A point in the map can only be feasible if the sequence of points leading up to it was feasible as well. This is a fundamental limitation of the application of in-sequence localization and therefore any dimension reduction in the map must take care not to destroy this order.

During this matching process, it is critical to discern the true values of $d(t)$ because localization errors could misinform the driver or the localization algorithm. Now observe the two time series shown in the upper-right hand side of Fig. 3.1.

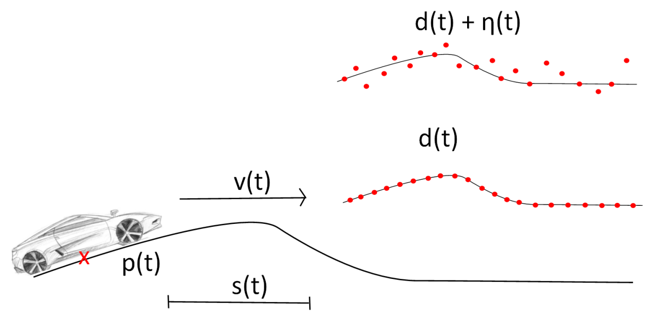

Figure 3.1. The effect of noisy sensors. An example from vehicle data collection.

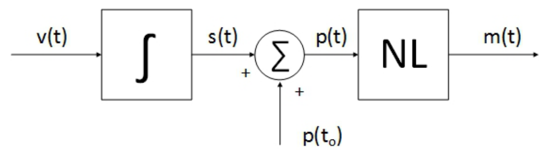

Figure 3.2. Vehicle Sensor Output Diagram

Note that a trend is clearly observed in the time series $d(t)$, but this trend is obscured when the noise is added in $d(t) + \eta(t)$. Thus it would be difficult to match $d(t)$ to $d(t) + \eta(t)$ without at least generating several false positive results. A more precise mathematical description of in-sequence localization is presented below.

Definition: Given a data set $\mathbf{D} = \{d_1, d_2, ..d_i\}$ that represents a complete sampling of a real world process and a small (i.e. $k \ll i$), noisy subsequence of length $k + 1$ collected at some later time T, $\bar{\mathbf{x}} = \{d_T + \eta_T, d_{T+1} + \eta_{T+1}, , d_{T+k} + \eta_{T+k}\}$, find the index j such that the subset $\bar{\mathbf{d}} = \{d_j, d_{j+1}, , d_{j+k}\}$ most closely represents $\bar{\mathbf{x}}$. Here most closely depends on the representation method chosen for $\mathbf{D}$.

If it is assumed that the data is noise-free, i.e. $\eta = 0\ \forall t$ and if it is assumed that there exists unlimited computational power and unlimited storage space, then the problem of in-sequence localization can be solved by brute force. This is the approach followed by many researchers who develop trajectory tracking algorithms [166]. While the state of the art may not be a fully brute force approach, it is nonetheless based on the ability to extract all possible trajectories and then evaluate their feasibility. For example, a sliding window of length $k + 1$ is used to extract all possible subsequences in the data set $\mathbf{D}$. Then each subsequence is evaluated against the query subsequence using the norm of the differences between the subsequence and the query. Finally, the subsequence that minimizes the particular norm is chosen as the most likely path, taking a step further to in-sequence localization, the last point in the subsequence is used to determine the location.

However, the computational cost of such an approach to in-sequence localization is high and grows with the size of the data set. To see this, suppose that the noise-free time series has a length i and the 1-norm is used for evaluation. Then the process takes one scan of $i-1$ subsequences each requiring $k(k+1)$ operations to complete. Thus as the size of the subsequences increases, and the size of the process time series increases, the order of operations approaches i^2.

In practical systems, neither unlimited computational power nor unlimited storage are available, nor will they be available in the foreseeable future because the pace of data collection matches or exceeds the growth in computing power. Thus as i and k grow, the practical burden becomes too great. In addition, while recent history has seen an explosion in the number of collected time series [14], this has come at the cost of increased noise characteristics of cheaper sensors. The new noisier data that is acquired using these sensors further increases the computational burden by requiring add-on noise mitigation algorithms or extended matching times that incorporate bigger time series.

To the best of our knowledge, the issue of noise mitigation has been historically cited in only a handful of papers [42, 46, 43, 12]. However, the recognition of the problem in large scale data applications is growing. In the vehicle localization domain, the authors of Stoyanov et. al. [16], Mullane et. al. [17], and Vemulapalli et. al. [12] discussed the effect of sensor noise on the building of localization maps. In the realm of air traffic control where the piecewise linear representation (PLR) is used to model aircraft motion data, Guerrero et. al. [18] discusses the limitations of PLR with respect to noise and and suggest improvements to the PLR algorithm handle noise. Lastly, dealing with optical data, Skauli [19] discusses the effect of sensor noise in storing and processing hyperspectral data that is recorded from optical sensors.

3.2 Dimension Reducing Data Representations

This section will present the data representation methods. These methods are typically applied to reduce the size of the stored data for the purpose of subsequence matching at a later point. In other words, these methods create possible sets of subsequences such that a similar subsequence can be identified. In this thesis the methods are applied to create a map of representations to be used for in-sequence localization. The aim of this section is both to describe each method and to test the representation method's robustness to noise. Section 3.2.8 will describe the testing procedure and the results for each representation.

3.2.1 Discrete Fourier Transform (DFT)

The discrete Fourier transform (DFT) is the first dimension reducing representation. Presented in 1993 by Agrawal et. al. [47], the use of the DFT as a dimension reduction tool signaled the beginning of the drive to find useful data representations. As published by Agrawal et. al., the DFT is used to represent an entire time series using just a few coefficients. The strength of the DFT as a data representation is that Parseval's theorem can be used to show that taking the DFT preserves the energy of the underlying time series. Then assuming that the majority of signal energy is in low frequencies, saving the first 3-5 coefficients means that most of the signal energy information is preserved.

In this chapter, the DFT is adapted for in-sequence localization by segmenting the time series into fixed window length intervals and taking the DFT within each window. As shown in table 3.1, the segmentation procedure begins by choosing the number of segments N_s into which the time series will be segmented. Then in each segment the DFT is performed and 5 coefficients are saved.

The DFT is a classical approach to data representation. In the case of repeated, high signal-to-noise ratio data, saving the first few DFT coefficients provides an adequate representation of the data that preserves most of the signal energy. However, in the case of corrupted data, preserving the first few coefficients can be problematic. This is because in the presence of noise, the saved coefficients should be those that correspond to the strongest, most robust, frequencies in the data. These are typically not known during the segmentation process.

Choose Constants:
Number of segments : N_s
Input Time Series Data, read length: D_i
Segment the data:
Divide the data in N_s intervals of L points
Discard the last set of points that do not fill an interval
For each interval:
Find the DFT of the data
Save 5 coefficients as the DFT representation
Store the representation

Table 3.1. DFT Segmentation Algorithm

3.2.2 Piecewise Aggregate Approximation (PAA)

One of the most intuitive data representations is the data mean. The data mean is a simple, easy to comprehend representation, and it is a dimension reducing representation because an arbitrary number of data points can be represented by a single number. However, representing an entire time series using a single number is not sufficient. To alleviate this problem, a sequence of interval means can be stored such that the resulting sequence is a pattern that represents the original time series. The sequence of interval data means is called the piecewise aggregate approximation (PAA) and it was first presented by Keogh et. al. [28] and Yi et. al. [29]. An example of a time series represented using PAA is shown in Fig. 3.3.

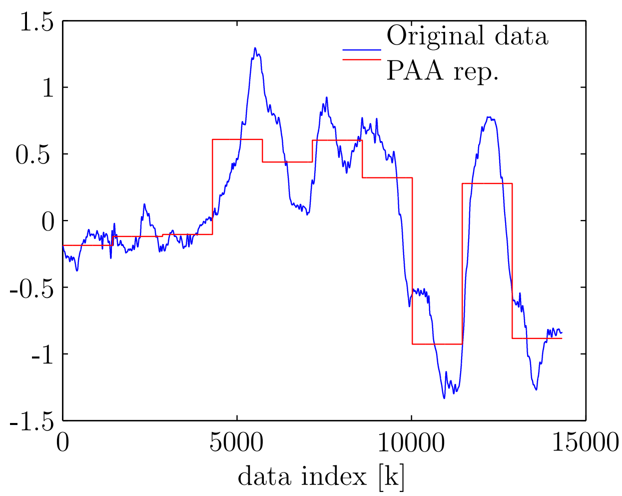

Figure 3.3. A sample PAA representation, representing a data set by 10 segments.

The procedure used to obtain a PAA representation is shown in table 3.2. First, a time series is input into the algorithm. The length of the time series, L, is recorded. Then the data is divided into N evenly spaced intervals where N is chosen by the designer. The number of points is obtained by rounding down the fraction L/N, and the last few points that do not fill an entire interval are discarded. Then data in the i^{th} interval is represented by its averages, m_i and the sequence of averages, $\hat{D} = \{m_1, m_2, \ldots, m_N\}$ is stored as the PAA representation.

Choose Constants:
Number of segments : N_s
Input Time Series Data, read length: D_i
Segment the data:
Divide the data in N_s intervals of L points
Discard the last set of points that do not fill an interval
For each interval:
Find the first and last data point indices, k_s,k_e
Find the data average: $\varphi_k = \frac{1}{k_e - k_s + 1} \sum_{k=k_s}^{k_e} d_k$
Store the representation
The PAA representation is $\Psi = \{\varphi_1, \varphi_2, \ldots, \varphi_{N_s}\}$

Table 3.2. PAA Segmentation Algorithm

PAA is an intuitive representation with a low computational cost that is very effective in reducing the size of data dimensions. However, with respect to in-sequence localization there are several significant disadvantages. First, the resolution of the representation is chosen by the designer. Thus PAA is a supervised algorithm and requires significant attention over a large database. Second, the average of an interval of data is the DC frequency information of the data; therefore all higher frequency information is assumed to be noisy and discarded. However, it has been recognized in the literature that higher frequency components carry significant information that should be used in the segmentation of a time series [33, 34]. Lastly, the regular segmentation of data may interrupt a significant underlying feature. For example, note that in Fig. 3.3 the first peak in the data is bisected when intuitively it should be captured as a whole.

3.2.3 Discrete Wavelet Transform Representation (DWT)

The next representation is the Discrete Wavelet Transform (DWT). The DWT, obtained using the Haar wavelet transform, can be thought of as the more general case of PAA. Chan et. al. [161] first introduced the idea of the DWT as a dimension reducing representation by choosing the 3 largest DWT coefficients to represent a time series. Similar to the DFT, the DWT is extended for in-sequence localization by segmenting the time series into fixed width intervals and representing each interval by its Haar wavelet transform.

The Haar wavelet transform for each interval is shown in table 3.3. It begins by segmenting the given interval data into consecutive, non-overlapping, 2-point intervals. The averages are computed for each pair, and then the differences for each pair are computed and divided by 2. The resulting vector is contains the Haar wavelet coefficients at resolution r. Now segment the vector of interval averages into 2-point intervals. Find the mean of the new set of intervals and the corresponding Haar wavelet coefficients. These Haar wavelet coefficients correspond to a resolution of $r - 1$. This procedure is iterated until a single average is found. This is the $r = 0$ wavelet resolution. The total number of resolutions determined will be the length of the time series L divided by 2.

For resolutions $r = D_i/2$*-1 :*
Segment time series into 2-point intervals,
Find interval averages,
Find wavelet coefficients by finding the average difference in the 2-point intervals.
For resolutions $r = D_i/2$*-2 : 0*
Segment the average vector from previous resolution into 2-point intervals,
Average the resulting intervals,
Find wavelet coefficients by finding the average interval difference.

Table 3.3. DWT Segmentation Algorithm

Analyzing the procedure to obtain the DWT, one can observe that the coefficients are built by a series of low pass operations. The underlying assumption is that high frequencies contain noise and that the low frequency information signal is sufficiently strong. While more frequency components are used than in PAA, the representation of the time series is not detailed until a large number of segments is used. For in-sequence localization, these issues are significant since the information carrying frequencies are unknown, and the desired representation should have as few segments as possible.

3.2.4 Symbolic Aggregate Approximation (SAX)

Building on the foundation of PAA, Keogh et. al. [54] introduced the symbolic aggregate approximation. In this representation, the data means are replaced by equi-probable symbols that correspond to a range of possible data means. The data mean ranges that correspond to each symbol are chosen such that each symbol has an equal probability of occurrence. An example of the SAX representation is shown below in Fig. 3.4 for $N = 10$ segments and 6 equi-probable symbols. The procedure to obtain a SAX representation is shown below in table 3.4.

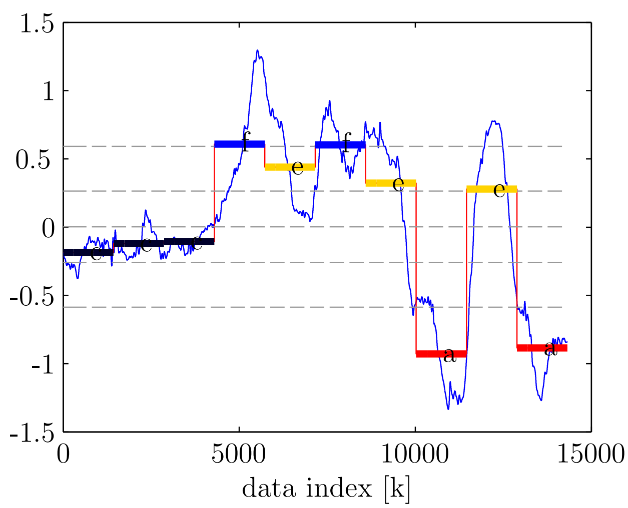

Figure 3.4. A sample SAX representation, representing a data set by 10 segments.

The first step in obtaining a SAX representation of data is choosing the number of segments into which the time series is to be segmented, N_s, and the number of equi-probable symbols, ϑ. Then a time series of data is input and normalized using the data mean, μ and the data standard deviation σ. The resulting standard normal distribution is subdivided into N_s regions and the region bounds are determined from the data. The data is then partitioned using the PAA procedure in table 3.2. Finally, each PAA mean, φ_k is translated into a symbol, $\hat{\varphi}_k$ according to the regions determined in the first step. The resulting sequence of symbols is the SAX data representation.

Choose Constants:
Number of segments : N_s
Number of symbols : ϑ
Input Time Series Data, read length: D_i
Determine the equi-probable region for each symbol:
Normalize the data by subtracting the data mean, μ, and dividing by the data standard deviation, σ: $\tilde{d} = \frac{d-\mu}{\sigma}$
Find S equi-probable ranges
Find the PAA representation of the data
$\Psi = \{\varphi_1, \varphi_2, \ldots, \varphi_{N_s}\}$
Assign the SAX symbol for each m_i
$\Psi = \{\hat{\varphi}_1, \hat{\varphi}_2, \ldots, \hat{\varphi}_{N_s}\}$

Table 3.4. SAX Segmentation Algorithm

The SAX representation is particularly well-suited to noisy data because comparing symbols does not require thresholding. However, with respect to in-sequence localization where the reference data set and the query data are not collected simultaneously, there are two problems. First, the normalization that is required to assign the symbols is problematic if the entire time series is not available. For example in vehicle localization the map data can be normalized, but the incoming localization data may not be of sufficient length to be normalized. Furthermore, no distinction is made of symbols that are near their respective probability boundaries. For example, in Fig. 3.4 the symbol “f” can easily be perturbed by a small amount of noise and miss-assigned during translation.

3.2.5 Chebyshev Polynomial Representation

Another method of representing time series is using Chebyshev polynomials [162]. The advantage of using Chebyshev polynomials is that they closely approximate minimax polynomials, which are used to minimize the maximum deviation between the polynomial and the model data. Because these polynomials minimize the maximum deviation, they improve the ability to select in between time series when matching. The Chebyshev polynomial representation is intended to describe whole time series. Here we extend again the approach to in-sequence localization by instead describing consecutive intervals of the time series and, in effect, localization sequences of Chebyshev polynomial representations.

The Chebyshev polynomial approximation begins by selecting the number of segments into which the time series is to be segmented, N_s. Then N_s equally sized intervals are extracted. For each interval the data is approximated using three Chebyshev polynomials,

$$\hat{d}_k = c_0 P_0 + c_1 P_1 + c_2 P_2, \tag{3.1}$$

where P_i are the Chebyshev polynomials and c_i are coefficients to be determined. The Chebyshev polynomials are approximated as [162],

$$\begin{aligned} P_0[k] &= 1 \\ P_1[k] &= k \\ P_2[k] &= 2k^2 - 1 \end{aligned} \tag{3.2}$$

and the coefficients are determined as,

$$\begin{aligned} c_0 &= \frac{1}{k_e - k_0} \sum_{j=k_0}^{k_e} d_k P_0[k] \\ c_i &= \frac{1}{k_e - k_0} \sum_{j=k_0}^{k_e} d_k P_i[k] \end{aligned} \tag{3.3}$$

The approximation from each interval is concatenated in the final representation, $\Psi = [\{c_0, c_1, c_2\}, \{c_0, c_1, c_2\}, \ldots]$. This procedure is summarized below in table 3.5.

Choose Constants:
Number of segments : N_s
Find representations:
for each interval find the constants c_i
using eqns. (3.2) and (3.3)
Store the final representation:
$\Psi = [\{c_0, c_1, c_2\}, \{c_0, c_1, c_2\}, \ldots]$

Table 3.5. Chebyshev Segmentation Algorithm

In contrast to other representations, the idea behind Chebyshev polynomials is to represent the data most faithfully to the original time series is a novel approach to data representation. From the point of view of in-sequence localization, this is not necessary. It has been observed in the authors' work that a few significant frequencies in the data dictate the robustness and performance of in-sequence localization algorithms [24, 167]. For this reason, a separate study would be needed to determine which Chebyshev coefficients are most useful in this application and for each unique time series to be indexed.

3.2.6 Piecewise Linear Representation (PLR)

Another intuitively simple representation that captures the trends in the data is the piecewise linear representation (PLR) [56]. This is the first representation in this chapter that chooses the segmentation points such that the mean square error of the approximation is minimized. For this reason, PLR is termed an adaptive representation in the data mining literature [15, 14]. An example of PLR can be seen in Fig. 3.5.

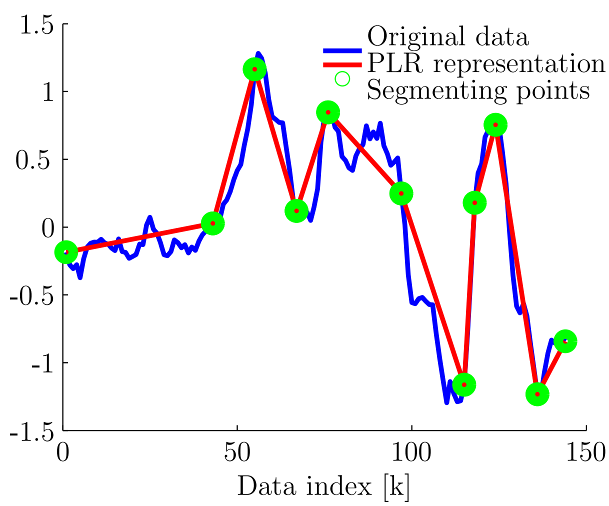

Figure 3.5. A sample PLR representation that describes a data set using 10 slopes.

The segmentation procedure for PLR, table 3.6, begins by segmenting the entire data series into consecutive 3 point intervals. Each interval is modeled as a slope between the first and last data point, and the mean square error (MSE) over the entire data series is calculated. Then one pair of intervals (ex. intervals 1 and 2) is merged and the slope is modeled from the first to the last point of the merged interval. The MSE of the estimation with one merged pair is calculated. The MSE is then calculated for all possible cases where one set of neighboring intervals is merged. From all merging possibilities, the merge that minimizes the MSE is chosen as the merge point. This procedure is iterated until only N segments remain. The resulting representation is the representation with the smallest mean square error given the number of segments specified by the user. By obtaining the minimum MSE representation, PLR addresses the fundamental problem with fixed-width segmentation, which is the inability to capture changes in the data dynamics.

Choose Constants:
Number of segments : N_s
Initial Data Segmentation:
Segment the data into $\Upsilon = D_i/3$ consecutive intervals
Approximate each interval by a slope
Segmentation Procedure:
While number of segments $> N_s$
For $u = 1$: Υ - 1
Merge segments u and $u + 1$, find new MSE
END
Choose the segment merge that minimized MSE
$\Upsilon = \Upsilon$ - 1
END

Table 3.6. PLR Segmentation Algorithm

PLR is a popular representation that has been tuned in some papers [18] to minimize the effects of noise. However, in the case of in-sequence localization, this type of adaptive segmentation is not possible because the query data is a subset of the entire time series and thus cannot be optimally segmented. In addition, segmenting time series with PLR is a computationally intensive process. For example, in the numerical simulations supporting this chater, it was found that at most 1400 data points could be evaluated across a set of possible segmentations within an 18 hour period.

3.2.7 Adaptive Piecewise Constant Approximation (APCA)

The major drawback of fixed window length segmentation is that changes in the data dynamics do not necessarily occur at regular intervals. To remedy this problem, Chakrabarti et. al. [163] created an adaptive segmentation approach termed the Adaptive Piecewise Constant Approximation (APCA). Given the number of desired segments, N, APCA uses the largest N Haar wavelet coefficients as segmentation points. In each segment, the deviation of the data mean from the whole time series mean ($r = 0$ Haar wavelet resolution) is used as the data representation. An example of the obtained representation is shown below in Fig. 3.6.

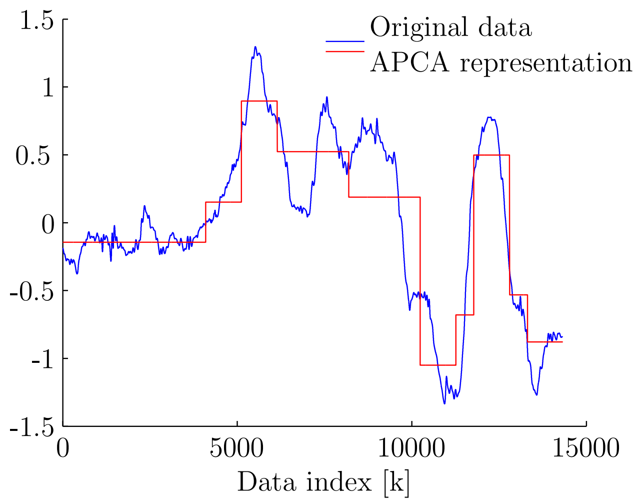

Figure 3.6. A sample APCA representation with 10 adaptive segments.

The APCA segmentation procedure, shown in table 3.7, begins by finding the Haar wavelet coefficients for all resolutions r. These resolutions are assembled in descending order in a triangular matrix. The K largest coefficients are chosen as the segmentation points. Then for each of the determined intervals, the mean of the data is compared to the resolution average $r = 0$. This corresponds to the average of the whole time series. The error between the time series average and the interval average is the data representation.

APCA is a data adaptive approach that captures the changes in the data behavior in a much more computationally efficient manner than PLR. However, in the application of in-sequence localization, APCA still suffers from the inability to optimally segment the query sequence. In addition, APCA has a much higher

Choose the number of intervals, N_s
Find all Haar Wavelet Coefficients
Choose the K largest coefficients to be segmentation points
Represent data as the error between the interval average and the time series average
Haar Wavelet Procedure:
For resolutions $r = D_i/2\text{-}1$:
Segment time series into 2-point intervals,
Find interval averages,
Find wavelet coefficients by finding the average difference of each interval.
For resolutions $r = D_i/2/2\text{-}2 : 0$
Segment the average vector from previous resolution into 2-point intervals,
Average the resulting intervals,
Find wavelet coefficients by finding the average difference in each interval.

Table 3.7. APCA Segmentation Algorithm

computational burden when compared to the more intuitive representations like PAA and SAX.

3.2.8 Additive Noise and Representation Fidelity

When evaluating the usefulness of data representations in the application of in-sequence localization, the key criterion is fidelity in the presence of noise. In particular, the data translation mechanism should be invariant in the presence of noise or at least have a close representation. This section evaluates the presented dimension reducing representation with respect to 12 dB of Gaussian noise. This noise was chosen for consistency with later Chapters in this thesis in which the noise was chosen to correspond to the type of noise observed in a mid-grade vehicle inertial measurement unit sensor [20, 23, 24, 167].

3.2.8.1 Testing Procedure

The procedure used in testing is shown in table 3.8. First a vector is created that holds all possible segment numbers to be evaluated. Then for each number of segments, a noise free representation is obtained and stored. The data is then corrupted with 12 dB of Gaussian noise and the noisy representation is formed. The noise free and noisy representations are compared, and any differences larger than 1 standard deviation of the noise are considered an error. The exception to this is SAX, which is not a numerical representation and can thus be evaluated in a "yes/no" fashion.

All representations are tested on four data sets: vehicle pitch data from State College, PA, USA; vehicle pitch rate data (the pitch derivate); random walk data; and random walk rate data. The vehicle data sets have a length of 14,000 data points, and the random data time series are 10,000 points long. The results shown in the figures below represent the mean number of errors across 100 trials for each of the testing data sets.

Choose segmentation method:
Segment the original time series (Ψ)
Corrupt the data:
Corrupt the data using 12 dB of White Gaussian Noise
Segment the corrupted time series (Ψ_η)
Compare the two representation:
$\mathrm{sum}(\lvert\Psi - \Psi_\eta\rvert > \sigma_\eta)$

Table 3.8. Representation Fidelity Testing Procedure

3.2.8.2 Results

The testing results are presented below in Fig. 3.7 - Fig. 3.14. In each figure, the x-axis represents the number of segments into which the time series was segmented. The y-axis shows the representation method, and the z-axis shows the probability of a translation error. Each plot represents the average rate of error across 100 trials for the corresponding data set.

From the plots one can see that with the exception of SAX that the data representation error rates are high and increase with the number of segments. This indicates that the number of incorrectly transcribed representations is very high, and that noise affects smaller segments more significantly. A notable exception to this is SAX, which represents data as a symbol corresponding to a pre-defined region of probability. The definition of a symbol across a range of values automatically accounts for some noise and distortion in the data. Despite this, in our experiments, SAX suffered from the lack of data normalization which is an integral step in the SAX representation process.

In addition SAX suffered from the high number of symbols that were used. In these experiments, 26 symbols were used in the SAX representation. The high number of symbols was necessary because in-sequence localization requires at minimum some level of uniqueness in the local sequence of symbols. Restricting the number of SAX symbols to a smaller number would greatly increase the resulting localization distance due to a large decrease in the symbol subsequence uniqueness.

While studying these figures it is important to also note the type of noise that is used in testing. Here white Gaussian noise corresponding to an overall signal to noise ratio of 12 dB was used. This means that representations that employ data averaging naturally perform slightly better because the standard deviation of the noise in the data is reduced by the square root of the number of averages. Thus is its expected that the PAA and SAX representations will do slightly better. Nonetheless, it is obvious from these experiments that the representation translation mechanisms are greatly affected by white Gaussian noise, and the results foreshadow the in-sequence localization results to be presented later because quick, efficient, and accurate in-sequence localization depends on the ability to discern sequences of representations in the map.

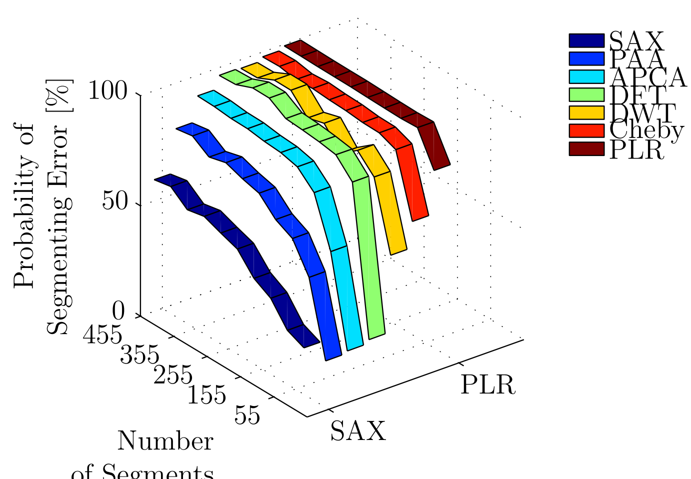

Figure 3.7. Fidelity of All Segmentation Methods in vehicle data with respect to 12 dB of Gaussian noise (data length = 1,500 pts)

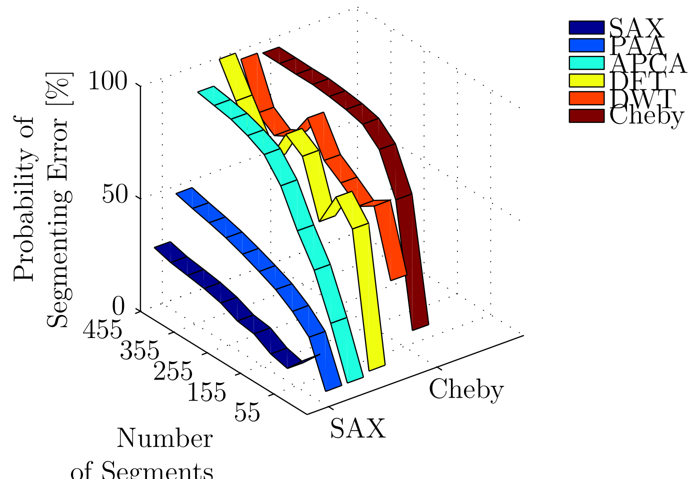

Figure 3.8. Fidelity of All Segmentation Methods except PLR in vehicle data subject to 12 dB of Gaussian noise (data length = 15,000 pts)

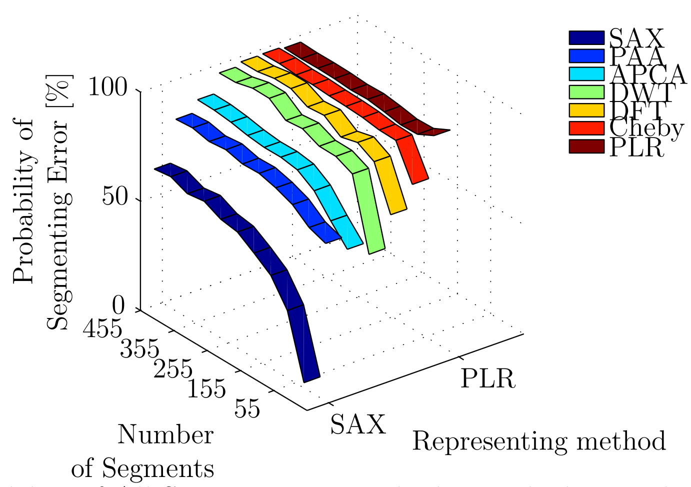

Figure 3.9. Fidelity of All Segmentation Methods in vehicle rate data with respect to 12 dB of Gaussian noise (data length = 1,500 pts)

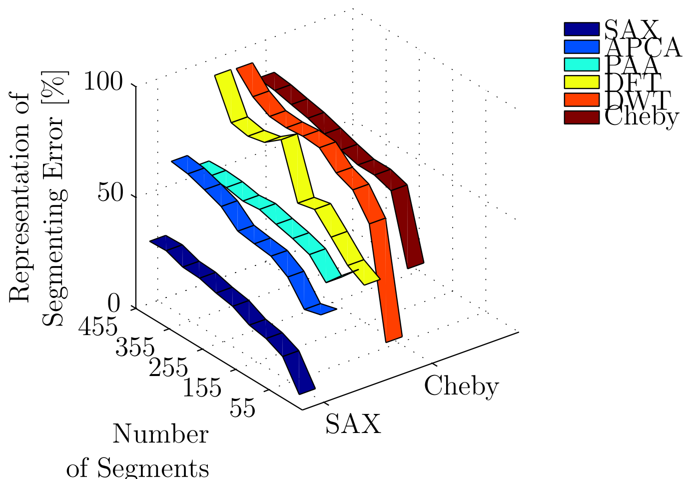

Figure 3.10. Fidelity of All Segmentation Methods except PLR in vehicle rate data that is subject to 12 dB of Gaussian noise (data length = 15,000 pts)

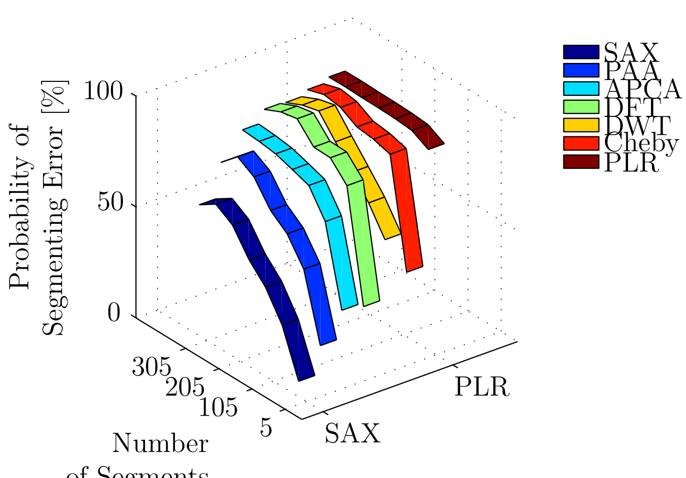

Figure 3.11. Fidelity of All Segmentation Methods in Random Data that is Corrupted with 12 dB of Gaussian noise (data length = 1,500 pts)

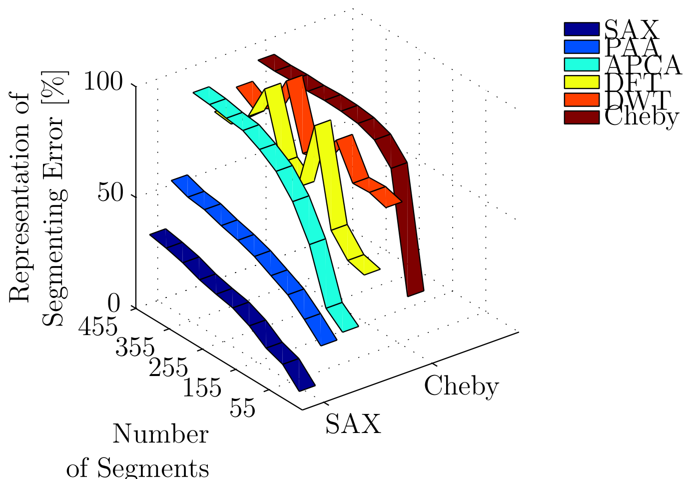

Figure 3.12. Fidelity of All Segmentation Methods except PLR in Random Data that is Corrupted with 12 dB of Gaussian noise (data length = 15,000 pts)

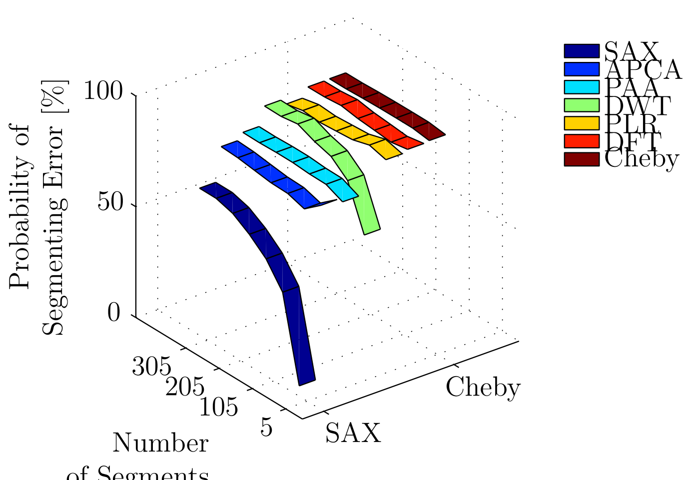

Figure 3.13. Fidelity of All Segmentation Methods in Random Rate Data that is Corrupted with 12 dB of Gaussian noise (data length = 1,500 pts)

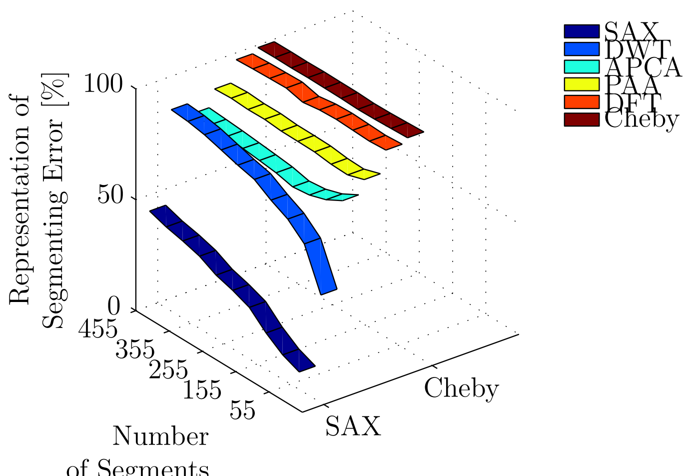

Figure 3.14. Fidelity of All Segmentation Methods Except PLR in Random Data Corrupted with 12 dB of Gaussian noise (data length = 15,000 pts)

3.3 In-Sequence Localization

The problem of in-sequence localization is fundamentally different from that of subsequence matching. Subsequence matching, the problem for which most dimension reducing representations are designed, is the matching a time series to either a stored time series of equal length or a type of previously described time series. This means means that in subsequence matching both the query time series and the pre-established database are available during the matching process. Features are extracted from the data to reduce the dimension of the data to simplify the computations necessary to evaluate a match.

In-sequence localization, on the other hand, is the matching of a fragment of a time series to a previously stored complete time series. Moreover, when a match is determined, the last data point in the time series fragment is used to estimate the final location of the data collection process. During in-sequence localization, new data is continuously acquired and each fragment of data that is received is immediately applied for localization. The goal is to quickly, efficiently and robustly identify the location of each new acquired information bit in the stored database.

Because in-sequence localization is carried out on a mobile platform, computational complexity of the data representations must be low, such that the computational burden during matching is minimized. Furthermore, because in-sequence localization is supposed to be fast and efficient the number of data points that must be acquired prior to beginning in-sequence localization must be minimized. These requirements constrain the number of data representation that can be used in this application, and they dictate that the number of data segments be large, such that the maximum segment size is acceptable as a starting point for localization. Lastly, small segment size is also desirable because it is (locally) unique sequences of represented intervals that lead to rapid localization, not large repeated representations. In essence, rapid, efficient and robust localization is achieved when the largest common subsequence in the representation is small.

There are two types of representations described in this chapter: fixed window length and variable window length (data adaptive). The remainder of this section will describe the performance of these representations for in-sequence localization using the realistic scenario of noisy incoming data. Section 3.3.1 describes the procedure for in-sequence localization using fixed window length representations. Section 3.3.2 will describe the localization method for adaptive data representations. Lastly, section 3.3.3, compares the performance of all data representations across found different data sets: vehicle data representing the terrain data collected by a moving vehicle for localization, vehicle data rate, representing the differences between neighboring terrain data points, random walk data generated using uncorrelated white noise, and the differences between neighboring random walk data points. Testing on each data set is performed for all possible starting points on the map and the results are averaged across 100 trials per starting point.

3.3.1 Fixed Window Length Representations

Of the the two types of data representations, the fixed width representations are much easier to implement for in-sequence localization because the number of data points to be acquired is the same regardless of the starting point in the data. The procedure to test in-sequence localization begins with the creation of a localization map. That is, a complete time series is segmented using a user specified representation method (PAA, SAX, DFT, DWT, Chebyshev polynomials). Then the time series data is corrupted using 12 dB of Gaussian noise. From among all interval starting points, a random query start point is chosen. The number of data points corresponding to one interval is collected and the new data is represented using the chosen method.

In-sequence localization begins when a localization map is built and an initial localization segment is collected. The localization segment is matched to each interval in the map, and the error is compared to the standard deviation of the corrupting noise. Errors that are smaller than the standard deviation are considered a match for the segment, and errors that are larger than the standard deviation are considered infeasible segments. After all map intervals are compared, the number of matches is counted. If the number of matches is one, then the interval is localized. If the number of matches is greater than one, then a second interval of data is collected and represented. The two-interval sequence is then matched for each possible two-interval sequence of segments in the map. The number of matches is counted, and the procedure is iterated until a single possible match remains. When a match has been determined, the match is evaluated to determine if it is a false positive, i.e. an apparent match with the wrong end segment, or a correct detection, a match with the correct end segment. The number of matching steps[2] is recorded to analyze the distance that was necessary for localization. This procedure described herein is summarized in table 3.9.

The testing procedure is iterated for 100 individual trials on each possible starting point in the map. This iteration allows the collection of probability of error rates and the average number of matching steps (number of segments collected prior to localization) for localization. The results of these experiments across the four specified data sets are further shown below.

Choose segmentation method:
PAA, SAX, DFT, DWT, Chebyshev polynomials
Create a noise-free localization map:
Segment the time series using the chosen method
Corrupt the data:
Corrupt the data using 12 dB of White Gaussian Noise
Choose a starting point:
Collect one interval of corrupted data and
represent using the method above
Localization:
While $num\ matches > 1$
For each possible segment in map:
Compare localization interval to map representation
Error $> \sigma_\eta$ means segment infeasible
Error $< \sigma_\eta$ means segment is a match
Count $num\ matches$
IF $num\ matches > 1$
Collect another interval of data, represent,
return to top of loop for matching

Table 3.9. Fixed Data Window Length In-Localization Procedure

3.3.2 Variable Window Length Presentations

The adaptive window localization procedure is much more intricate than the procedure described above because regardless of the start point in the map, in-sequence

[2]Number of segments collected prior to localization

localization requires matching to begin at the smallest possible segment. The modified procedure is shown in table 3.10 and described below.

First a method of representation is selected. From the methods described in this chapter, PLR and APCA are adaptive methods, capturing the changes in data behavior. Then a time series is segmented. To simulate the online acquisition of noisy data, the time series is corrupted with 12 dB of Gaussian noise and a random start point is selected. This start point must still be at the beginning of the map segments. The map is then scanned for the smallest possible interval size to determine smallest number of data points to be initially collected.

The initial data segment is collected and represented. The resulting representation is then matched to all segments of the smallest possible size in the map. If a single match is found, then the localization procedure is complete. If more than one match is found, then from among all possible segments, the next smallest segment subsequence is determined. This number of data points is acquired and represented. The new two interval sequences matched to the map to all 2 interval sequences of the same segment sizes. If any are feasible, a third segment is acquired and matching continues until a single segment is found.

However, if no matches are found in the first interval, several more data points are acquired that correspond to the next largest interval size. If any matches are found the procedure above is iterated. If no matches are found, then additional data is acquired to the next interval size. This type of exhaustive testing continues until all possible matches are tested for agreement and only a single feasible match remains. The difficulty in this approach is maintaining flexible trajectories that can change as paths are invalidated and new segment sizes are determined. The matching procedure for variable interval window length is summarized below in table 3.10.

3.3.3 Results

This section reviews the results of in-sequence localization using both fixed width and variable window width representation methods. There are two types of figures. The first is the number of matching steps needed until one match was left in on the map. These figures show the average distance across all possible map starting

Choose segmentation method:
Choices are: PLR or APCA
Create a noise-free localization map:
Segment the time series using the chosen method
Corrupt the data:
Corrupt the data using 12 dB of White Gaussian Noise
Choose a starting point:
Collect a number of data points corresponding
to the smallest segment size in the map
and represent using the chosen method
Localization:
match the newly acquired segment
to all segments of the same size
while *nummatches* > 1
Among all matches,
find the smallest second segment size
acquire data and represent as next segment
Count number of matches
Repeat while *nummatches* > 1
for all possible segment lengths

Table 3.10. Variable Data Window Length In-Localization Procedure

points [3] to localization. There are four separate figures, Fig. 3.15, Fig. 3.17, Fig. 3.19, and Fig. 3.21 showing the necessary distance to a match for all four data sets. In each plot, the x-axis shows the number of segments in the map, the y-axis shows the encoding method, and z-axis represents the number of matching steps.

The second type of plot is the plot of correct detection. That is, following the determination of a match, the final segment is evaluated to determine whether the correct location has been identified. The correct detection rate, or the percentage of trials which resulted in a correct localization, is shown with respect to the number of segments into which the time series is segmented and the segmentation method. The results across all four data sets are shown below in Fig. 3.16, Fig. 3.18, Fig. 3.20, and Fig. 3.22.

The results shown in this section demonstrate two facts about using representations designed for subsequence matching for in-sequence localization. First, the relative uniqueness of sequences of representations is high. This is evidenced by

[3] 100 trials per starting point

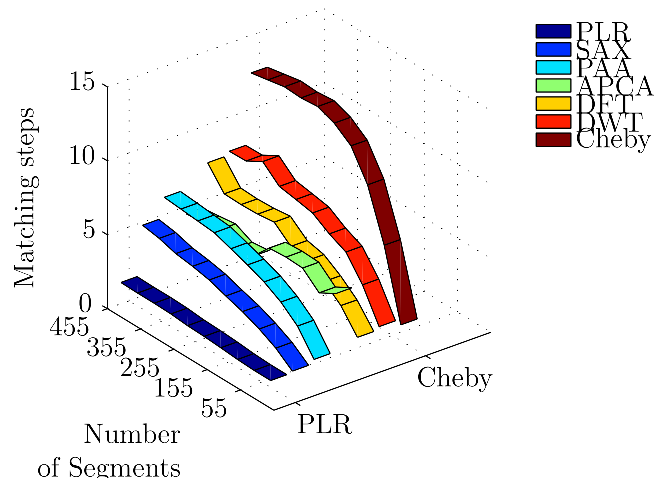

Figure 3.15. Average Number of Matching Steps During In-Sequence Localization in Vehicle Data

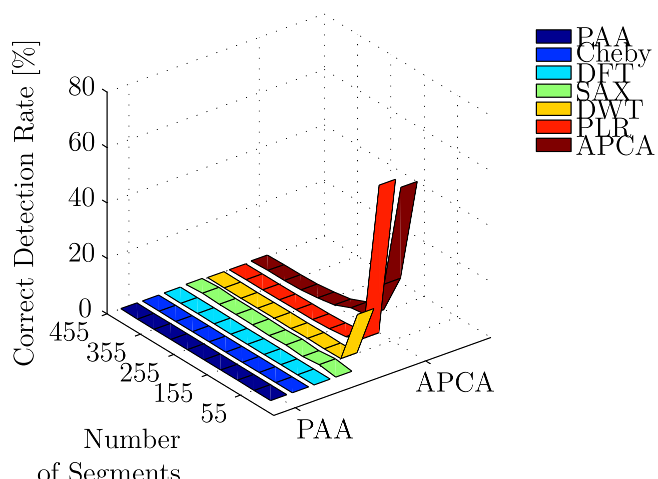

Figure 3.16. Correct Detection Rate During In-Sequence Localization in Vehicle Data

the rapid convergence of the majority of algorithms - less that 30 steps in the majority of cases. Second, despite converging rapidly, the algorithm is usually wrong, resulting in correct detection rates less than 10% for the majority of representations. Thus the presented data representations are significantly susceptible to the effects of noise.

A notable exception to the trends stated above are the variable width representation methods - PLR and APCA. Taking into account key points in the data as is the case in these representations, results in faster matching times and increased correct detection rates of up to 30%. Still, even for these representations increas-

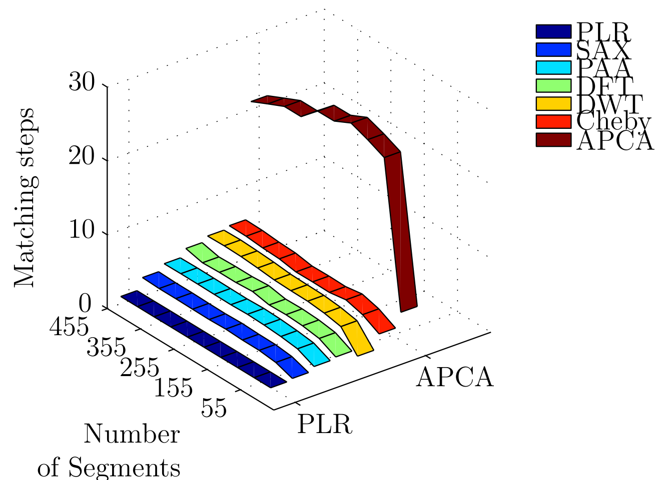

Figure 3.17. Average Number of Matching Steps During In-Sequence Localization When Using Vehicle Data rate

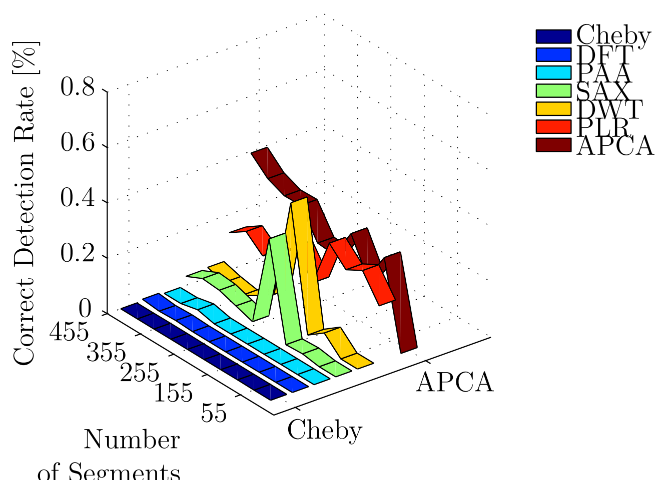

Figure 3.18. Correct Detection Rate for In-Sequence Localization in Vehicle Data Rate

ing the number of segments dramatically reduces the performance of in-sequence localization which suggests that there are relatively few important points to be used for in-sequence localization on any data set.

3.4 Discussion

The results in the previous section demonstrate the fact that the seven current reviewed methods have serious shortcomings for in-sequence localization. This is not surprising, since the representations reviewed in this chapter are designed for

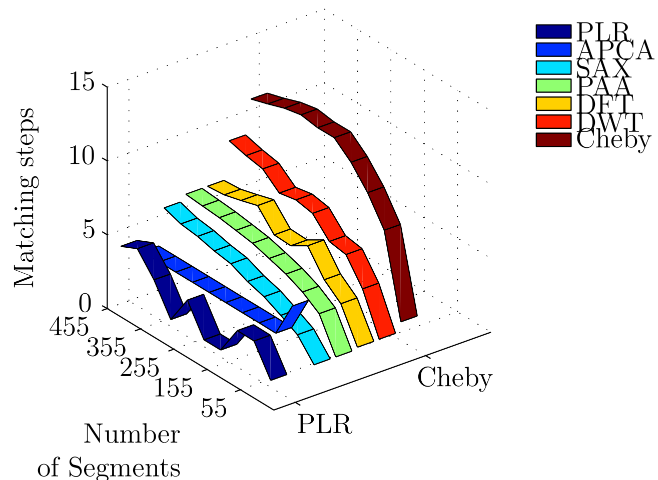

Figure 3.19. Average Number of Matching Steps for In-Sequence Localization in Random Data

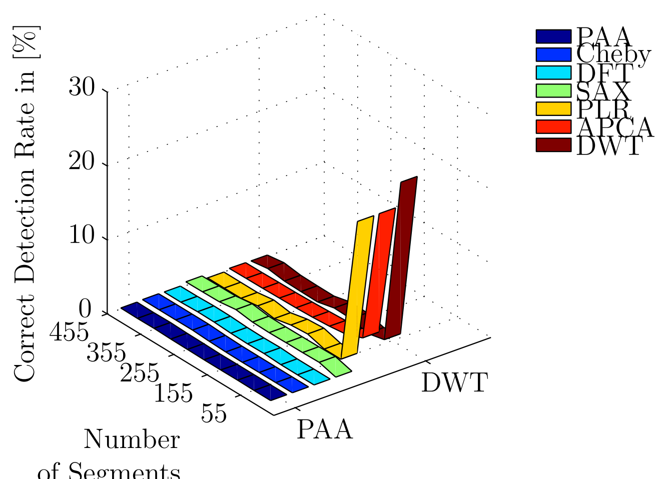

Figure 3.20. Correct Detection Rate in Random Data for In-Sequence Localization

subsequence matching and thus do not account for the time component of the data. Because of this, when the representations are tested for in-sequence localization, an application that inherently requires a tracking of the temporal nature of the data, the representations fail. One important observation that can be made from the results is that taking into account the significant points in the data where the behavior of the underlying data generating process changes is an important step for any in-sequence localization algorithm. Thus any algorithms that are developed for this purpose should be adaptive, and capable of capturing such changes.

In addition, all representation approaches above lack the basic requirement

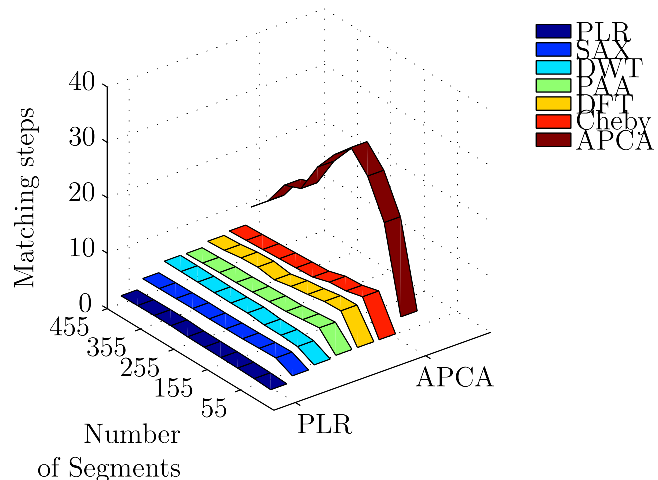

Figure 3.21. Average Number of Matching Steps When Performing In-Sequence Localization in Random Rate Data

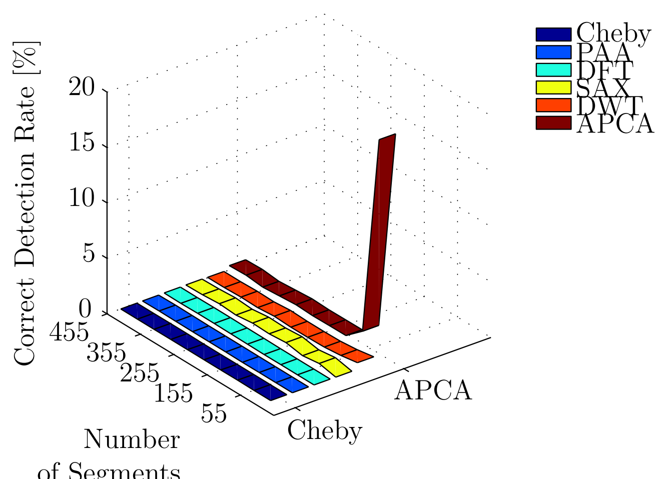

Figure 3.22. Correct Detection Rate in Random Rate Data During In-Sequence Localization

for in-sequence localization that localization can begin anywhere in the segment. Because of this the localization procedure needed increasingly smaller segments to minimize the starting point error. This resulted in worse performance in the presence of noise, because fewer time points were used to mitigate the effects of noise.

Recent work that is described in this thesis has focused on mitigating some of the negative effects of additive noise (see later chapters or the publications [20, 23, 24, 22]). In fact this work has shown that by taking into account the temporal nature of the data, and some characteristics of the noise greatly improves the results. We have shown that using this extra information, the problem of converging to a correct location can be reformulate from that of whether the algorithm will correctly converge, to what is the localization distance, and the data decimation given a set of sensor noise characteristics.

Other authors have sought to do the same based on the existing data representations. For example, Guerrero et. al. [18] build on the foundation of PLR to improve its performance. The work is based on airplane tracking data used to improve flight controller information in Europe. By taking into account the predictable patterns of motion of the airplane, Guerrero was able to show significant improvements in matching using PLR in the presence of noise.

3.5 Conclusions and Future Development

This chapter introduced the problem of in-sequence localization by testing seven published dimension reducing representations. The testing was performed in two steps. First, the representations were tested to determine the fidelity of translation in the presence of sensor noise. Second, the representations were tested in noisy data and in the application of in-sequence localization. The results showed that published representations do not adapt well to noisy data, and lack the ability to track sequences that are temporal in nature. Because of this the published representations perform poorly in the problem of in-sequence localization.

Subsequent chapters will seek to develop a new data representation that is specifically targeted to the application on in-sequence localization. At first the representation will be tailored to meet the demands of random starting location, retaining the sequential information of the data, and minimizing the computational complexity of localization. Then in subsequent chapters we will address the noise observed in the data and scaling the problem.

Chapter 4

A Dynamics Discontinuities-Based Deterministic Data Structure Creation Algorithm for In-Sequence Localization using 1-Dimensional Time Series

The previous chapter introduced the problem of in-sequence localization and evaluated the performance of representations published for the application of subsequence matching in the new problem (in-sequence localization). The poor performance of the representations motivates the need to create a targeted representation that takes into account the requirements of in-sequence localization. This chapter introduces linear models as one possible method of storing data patterns for this application.

These linear models are structured in a tree-like fashion to simplify the online localization process. During localization, using linear models eliminates the need for the computation of an agreement metric by instead comparing each model output to the modeling error bound. Agreement or disagreement with this bound determines the segment feasibility.

Thus the introduction of linear models is advantageous because it reduces the complexity of both the mapping and the localization mechanisms while enabling the creation of simpler noise mitigation schemes. When compared to prior published work (by other authors) that uses linear models, this chapter uses models of order greater than 2 that more faithfully describe the data to preserve a greater portion of the information in the data. Preserving this information is the key to eliminating the matching metric during the localization process, and therefore the key to reducing the computational complexity. In addition using linear models allows the approach to be easily extensible to streaming data, large datasets and multiple dimensions.

Some of the preliminary work for this chapter was presented at the 2012 Penn State College of Engineering Research Symposium where it was awarded the second place paper award [21]. Further preliminary work was presented at the 2012 IEEE Conference of Decision and Control [20]. A full manuscript containing this work has been published in IEEE Transactions on Intelligent Transportation Systems [22].

4.1 The Need for New Vehicle Localization Approaches

The motivating application for the problem of in-sequence localization is vehicle localization. This is a critical area of research because of the rapid development of advanced vehicle safety systems, navigation systems, and intelligent transportation systems capable of optimizing traffic patterns, tracking resources and identifying infrastructure problems as they arise. All of these systems depend heavily on the ability to localize a vehicle and, in many cases, are significantly more effective with meter-scale resolution of the vehicle's location. In addition, any safety system must be highly reliable to prevent potential accident causing failures [6].

The Global Positioning System (GPS) has been the standard-bearer for vehicle localization for some time. Current single-frequency GPS receivers can achieve a minimum error of about 10m [7, 8]. Achieving meter resolution requires a two-frequency GPS receiver, which is currently cost prohibitive for consumers. Furthermore, even with the three independent global positioning systems, the U.S.'s

GPS, the European Gallileo, and the Russian GLONASS, there are still geographic locations that will experience insufficient coverage or multi-path effects due to signal occlusions [9, 8]. These situations are encountered sufficiently frequently on roadways to necessitate the development of alternate localization technologies.

To address these issues, manufacturers have focused on developing systems that augment the GPS position estimate. The augmentation is typically performed using a combination of vehicle sensors, road maps, and motion models. Regardless of what type of augmentation is performed, each augmentation strategy is dependent on vehicle sensors that provide information for the vehicle maps and models to estimate short term position changes during GPS drop-outs. However, costs can increase significantly when including additional sensors. This is particularly true as the field of research moves to visual odometry sensors such as cameras and LIDAR [168, 169, 170, 171].

Sensors that have proven to be reliable for localization and navigation include steering encoders, odometers, wheel tachometers, and inertial sensors. Published results have shown the first three are capable of accurately tracking a vehicle's location [114, 118]. In addition, recent work by Jo et.al. [172] demonstrated the fusion of these sensors with low cost GPS for effective localization. Inertial sensors in the form of inertial measurement units (IMUs) have been previously used in commercial and military-grade applications but suffer from integration errors that grow unbounded with respect to operation time. Nonetheless, many IMUs are sufficiently accurate for localization during brief dropouts [173].

Inertial measurements are particularly attractive when augmenting GPS data because they provide a self-contained source of measurements that is not susceptible to the environment. This complements the attributes of the GPS and creates a system more robust to environmental factors such as signal blocking and vehicle condition [6]. To address IMU output drift, the sensor output can be transformed into a set of features that neutralize integrative error. These features are stored in a reference map that is compared to new data during vehicle travel. An example of this is the work from Penn State that uses vehicle pitch to measure road grade, a disturbance measured in the vertical IMU's measurements. In this work, new features are correlated to the previously stored map [127, 119, 137, 146, 138, 120]. The use of reference maps is the main limitation in using IMUs because it requires substantial computational power and storage to support the map-matching [122] infrastructure. Therefore, new approaches to the creation and structuring of reference maps are necessary to improve the viability of using IMU data for localization when augmenting GPS, with map and feature representation research being a key area of ongoing research.

To this end, this chapter presents a dynamical model-based approach to the problem of localizing a road vehicle using inertial measurement data (see Fig. 3.1). In particular, the method described herein uses vehicle pitch data in a self-contained dead reckoning (DR) approach that does not require the addition of a GPS signal. Creating self-contained approaches is useful, even if used in conjunction with a GPS, because it creates both hardware and software redundancy in the localization process and thereby increases the reliability of the overall navigation system.

4.1.1 Problem Formulation

Previous localization work has addressed the implementation problems by either setting the localization problem in the probabilistic domain; using tools such as the Kalman filter, the Bayes filter, or the particle filter to obtain the location estimates; or by evaluating map location using a comparison metric. In contrast, this chapter chooses to frame localization as a deterministic problem by expressing the road data using linear models. Linear models address the problem of data compression by expressing large portions of the data using a small number of model coefficients. During localization, linear models simplify computations by eliminating the need for metrics to match incoming data. Instead classically defined error bounds can be used to define data agreement. Relative to the feature matching approaches, this further reduces the complexity of the transformations performed on the observed noise and simplifies noise mitigation algorithms.

Vehicle pitch, as recorded by the IMU, is the road grade filtered by the wheelbase of the vehicle. The collection of road data is illustrated in Chapter 3, Fig. 3.1. Typically during localization the collected data includes the variables $v(t)$, $s(t)$, and $m(t)$, and the localization problem is reduced to identifying the position $p(t)$. The challenge of localization can be further reduced to determining initial position constant $p(t_o)$.

In this work, the localization of the vehicle is accomplished by a two-step process. First, a specially instrumented test vehicle collects position and pitch road data[1]. This data is compressed and stored as a road map for use in another vehicle. The compression and storage of the data is performed by extracting linear models, which represent patterns from the data that are robust to noise. When a different vehicle travels along the same road, the data observed by its sensors are compared to the predetermined map. Regions of the map are continuously eliminated until the correct vehicle location is identified. This approach to localization is referred to as the map-matching approach.

4.2 State of Research

4.2.1 Sensor-Based Localization

The three sensors commonly used for DR localization are the odometer, the wheel tachometer and the IMU. A series of review papers can be found in [113, 114, 6]. Perhaps the simplest method of navigation is odometry. To keep track of odometry, manufacturers install a sensor that records a pulse each time it passes by a specific point on the wheel. The number of pulses is then multiplied by a scale factor that is related to the wheel diameter, tire pressure, temperature, and any environmental factor that changes the wheel radius. References citing the use of odometry and error-correcting techniques in vehicle localization can be found in Abbott and Powell [114] and also within the mobile robotics community [115, 116].

A similar set of sensors are the wheel tachometers. These sensors are typically employed in anti-lock brake systems (ABS) to detect the differences in wheel speeds. Using the same framework, the difference in wheel speeds can reveal the direction of the vehicle. Like the odometer, the critical piece of knowledge necessary to make use of velocity encoders is wheel radius. In Carlson et.al. [117, 118] velocity encoders are used in a dual GPS/DR system. Here while GPS is available, the signal is used to provide a heading and a wheel radius estimate. When GPS becomes unavailable, wheel tachometers are calibrated to provide heading

[1] The instrumentation of the vehicle is described in detail in section 4.3.1.

information.

Odometers and velocity encoders suffer from well-known sources of terrain error such as wheel slips, uneven road surfaces and skidding. In addition, these same sources suffer from vehicle structure errors such as wheel diameter changes, wheelbase uncertainty and low resolution of the encoding sensors. These errors have been observed and characterized both in the vehicle navigation community [114, 118] and the mobile robotics community in Borestein and Liqiang [115].

In contrast to odometry and tachometers, gyroscopes and accelerometers in IMUs measure rates of acceleration and rates of change in displacement of the vehicle. A typical IMU consists of three gyroscopes and three accelerometers, each directed in one of the axes of motion. To obtain vehicle heading and velocity, the IMU outputs are integrated in each direction. This integration leads to the cumulative growth of small bias and noise sources in the IMU as a function of the operation time. These noise sources will be further discussed in section 4.2.1.2. However, because IMUs are self-contained and thereby not susceptible to the vehicular and environmental noise sources, IMUs are a good complement to other vehicle sensors and GPS [119, 120]. For this reason, GPS and IMU data is frequently integrated in augmented systems [121, 6, 174].

4.2.1.1 Map-Matching

The DR localization approaches described above can be used to estimate vehicle location given a known starting point for the vehicle. However, if the location is unknown, then the vehicle position can be estimated through either triangulation using active beacons (the GPS is a form of active beacon) or map-matching using a pre-existing reference map [122, 123, 124, 175, 176]. A third alternative is under development where building the reference map and localization is carried out simultaneously [139, 177, 140].

The linear modeling approach to reference map creation and localization in this chapter combines DR using inertial measurements and map-matching. A state of the art review of map-matching is found in Quddus et.al. [124]. The author subdivides the field of map-matching into early techniques that take into account the road geometry, such as arcs and lines; topological approaches, which take into account road geometry and the interconnection between each geometric feature;

probabilistic approaches that assign a region of error within which a likely road segment is found; and more recent approaches termed advanced map-matching using Kalman Filters, fuzzy logic, particle filters, etc. The references therein provide an in-depth look of the field of map-matching.

As a general outline, the map-matching process begins by extracting or observing features from the data, Smith et.al. [125]. The choice of features during the extraction process has a pivotal role in the subsequent localization performance of the algorithm. Early algorithms use arcs and lines as features that modeled the road shape [122, 126]. In advanced algorithms, the features are extracted through a non-linear transform that is used to identify the most noise-robust portions of the data. A few examples are local extrema, geometric features such as arcs and lines, or geometric beacons. Further examples can be found in the literature on pattern matching and data mining [127, 128, 129, 130, 131].

Then, during localization, the features are re-extracted from new vehicle data. The location is determined by comparing these features to the map. The most common comparison approach is the threshold. Here a distance is computed from the road data to each possible map location. An empirical threshold is used to determine the correct segment. More advanced methods of matching include those presented in the robotics community where the probability of the vehicle's location and of the detected features is calculated using Kalman filtering in a formal framework methodology established in [132, 133, 134], or a Bayesian comparison approach such as the one used by Levinson and Thrun [135], or a particle filtering approach in Törnqvist et.al. [136].

The motivation to develop advanced map-matching algorithms stems from the need to handle increasingly complex road networks that easily overwhelm earlier algorithms. Unfortunately, to the best of our knowledge, there does not exist a unifying publication that elucidates the limitations of each map-matching approach in the vehicle navigation community. In contrast, several review papers in the data mining community have concluded that more work is necessary to design efficient algorithms to handle large databases [14].

Choosing the features and the extraction process plays a pivotal role in the subsequent performance of the algorithm. Previous research in our own group has focused on IMU pitch data for localization [10, 137, 138, 119, 120, 127]. In particular, Dean [10] demonstrated a particle filter-based approach using inertial measurements. A more noise robust approach was presented in Vemulapalli et.al. [137] where an optimal filter was derived to process the data prior to extracting extremum features. Combining previous work, the work by Katetotad et. al. [138] employed both the particle filtering and feature-based approaches to allow localization and tracking of a vehicle over a greater area.

4.2.1.2 Noise Characterization of the Inertial Measurement Units

Recent decreases in the cost of Micro-Electrical-Mechanical Systems (MEMS) IMUs have contributed to a rise in the number of publications citing augmented GPS/IMU systems [141, 119, 120, 142]. Given this development and the potential for further research in the future, it is important to understand how sensor limitations affect the localization approach. In particular, El-Sheimy et.al. in [143] characterize IMU noise in terms of Allan variance and power spectral densities. The same authors then offer approaches to mitigating IMU noise [144]. A MEMS specific analysis is demonstrated by Aydemir and Saranli [145], while the work by Jerath and Brennan [146] evaluated the noise performance of several grades of IMUs.

Extensive research has also been presented on the calibration of IMUs [147, 148, 149, 178]. An in-depth discussion of IMU noise components is shown in section 4.4.3. The described noise is then used to corrupt the IMU data for localization simulations.

4.3 Algorithm Description

4.3.1 Road-Map Model Extraction

The first step in our approach is to collect road pitch data using a specially instrumented vehicle. This vehicle has been equipped with the Honeywell HG1700 IMU, which is mounted to the console near the vehicles center of gravity between the driver and passenger seats. The vehicle pitch data is estimated using both the longitudinal gyroscope and longitudinal accelerometer readings and is internally filtered in the IMU using a factory-integrated Kalman filter that reduces the effects of IMU noise.

The data is then modeled using a set of AutoRegressive models with an eXogenous input (ARX) [179], that describes non-overlapping segments of the data. ARX models are particularly suitable for this application because the model output is a prediction of the incoming data. In this application the exogenous input is the error between the estimate and the collected data; it is assumed that this error is bounded by a bound ε. The general form of an ARX model is

$$\begin{aligned} d[k] &= c_1 d[k-1] + ... + c_N d[k-N] + e[k] \\ &= \mathbf{A}\,\omega[k] + e[k], \qquad |e[k]| \leq \varepsilon \end{aligned} \tag{4.1}$$

where k is the data index, $d[k]$ represents the current sample of data, $\mathbf{A} = [c_1, ..., c_N]$ is a vector that contains the coefficients of the linear model of order N, the vector $\omega[k] = [d[k-1], ..., d[k-N]]$ contains the previous N samples of data, i.e., the so-called regressor vector, and $e[k]$ is the model output error bounded by the above mentioned bound ε.

Models are extracted using a Greedy algorithm developed in [180]. This algorithm simultaneously obtains the ARX model coefficients and breaks the road data vector into non-overlapping segments. The algorithm is shown in table 4.1.

The algorithm begins at the $(N+1)^{th}$ data point, labeled k_0. Starting at this initial data point, the algorithm searches for the largest interval for which it is possible to obtain a single ARX system that satisfies the error bound for every point.[2] Once this is not possible, a transition is declared indicating the end of the model fit and start of the next model; the corresponding data index is labeled τ_0. Henceforth, τ_0 is called a transition point, and it is the point at which one model segment ends and the next begins. The model-fitted segment is removed and this process is repeated until the final data point is reached, i.e. k_{max} is reached. The resulting set of data segments spans the values of consecutive transition points: (d_0, τ_0),(τ_0, τ_1), etc.[3] The optimality of this algorithm is described below in proposition I. This proposition was proven in [180].

[2] The search itself is performed solving a linear feasibility problem.

[3] The model coefficients that correspond to each segment are not unique since multiple sets of coefficients may result in a model that satisfies the error bound.

Greedy Algorithm
Initialize Constants:
model order : N
precision variable: ε
segment index: $n = 0$
initial loop index: $k_0 = N + 1$
first "transition point": $\tau_0 = k_0$
Algorithm Loop:
FOR i = k_0:k_{max}
Find a vector $\mathbf{A}_n$ such that:
$\mathcal{F} : \{\lvert d[k] - \mathbf{A}_n\, \omega[k]\rvert \leq \varepsilon \;\; \forall k \in [\tau_n, i]\}$
IF $\mathcal{F}$ is infeasible
Store $\mathbf{A}_n$ from index i-1, set the data index bounds:
$I_n = [\tau_n$,i], iterate the segment index: n = n + 1,
Store the transition point: τ_n = i
END IF
END FOR
$I_n = [\tau_n, k_{max}]$ and $\tau = \{I_j\}_{j=0}^{n}$
Return n and τ

Table 4.1. Optimal Greedy Algorithm for ARX Model Identification

Proposition I: Given a bound on the error, ε, and a model of order N, the algorithm described in Table 4.1 breaks the collected data set into the smallest possible number of segments.

The segmentation of the data set is illustrated in Fig. 3.1. In this figure, the horizontal axis represents the distance in meters from the initial map position. The vertical axis represents the angular pitch rate output of the IMU. In each sub-figure the solid line represents the query data, the solid line with circle markers displays the model output, and the dashed lines represent the error bounds. In each case the model is only valid for a portion of the plotted data. For example, the first sub-figure shows a model valid from 300 m to 500 m. Then the next plot shows the model that belongs to the segment from 500 m to 600 m. Lastly, the third plot shows a model that is valid for the remaining distance. Note that each model is within its error bound for the given segment. Outside of this segment the model may or may not agree with its bound. It is the unique sequence of these models, even if they themselves are not unique, that will be used in the localization algorithm development.

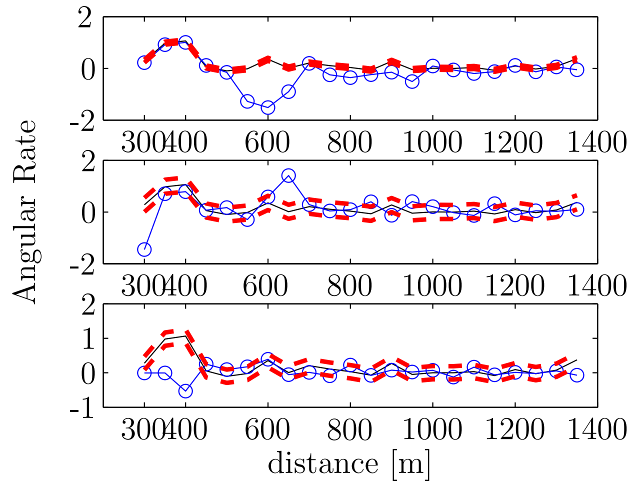

Figure 4.1. ARX Model Fit Demonstration

4.3.2 Locating a Vehicle using the Extracted Models

Vehicle localization can be subdivided into two distinct problems: locating the vehicle without *a priori* knowledge of the start location and tracking a vehicle given knowledge of its initial position. The set of ARX models extracted in the previous section can be used to perform both functions. At first, segments are continuously eliminated until a single feasible segment remains. This segment represents a possible set of vehicle locations. Then, when a transition between models is found, the precise location of the vehicle is detected. From this point forward, the vehicle's location can be tracked through validation of the current segment and detection of future transition points. The iterative process by which this is accomplished is described in table 4.2. In this algorithm the variable "L" is set to one.

Similar to the Greedy algorithm in Table 4.1, the localization procedure in Table 4.2 begins with the collection of $N + 1$ pitch data points. The data index is set to $k = N + 1$, and the acquired data is sequentially tested in each of the extracted linear models, denoted by the index m. The output error of the models is compared to their error bound, ε_m. More precisely, this comparison is shown in equation (4.2).

If for a given set of data the model error is smaller than the bound, then the current segment is labeled as a possible, or feasible, set of vehicle positions; otherwise the segment is labeled as infeasible. At each time step, only segments that were previously labeled as feasible are tested. Thus, under some assumptions to be discussed below, the number of feasible segments monotonically decreases to one after some iterations.

$$|d[k] - \mathbf{A}_m \omega[k]| \gtrless \varepsilon_m. \tag{4.2}$$

Two critical assumptions must hold to facilitate the localization of the vehicle. First, the pitch map must not be periodic to within an epsilon bound equivalent to the model error. For real-world pitch maps that have been measured to date by the authors, roughly 10,000 km or 6,000 miles, this assumption is quite valid. Second, the vehicle must travel a distance sufficiently great such that there exists a unique observed sequence of models from among all possible sequences of models on the map. The length of necessary travel distance is dependent on the starting location of the vehicle on the map.

As an illustrative example, consider a vehicle that begins traveling on a path that contains two hills and otherwise flat road in between and around the hills. Suppose that the hills are similar in their pitch profiles. Then no single segment of the map alone is sufficiently unique to identify the location of the vehicle. For a small travel distance the algorithm would have at least two possible feasible paths during localization. In this example travel distances smaller than the distance between the hill peaks would result in this localization ambiguity.

However, if the vehicle travel is extended to at least the distance between peaks of the hills, then there is only one unique profile which can be identified. The unique profile in this case is the flat road ahead of the vehicle followed by two hills and then flat road again. Similar to this example, each time a vehicle begins travel, there exists some distance within which a unique localization sequence of models can be found.

4.3.2.1 Model Transitions

If the vehicle is traveling inside of a model segment and a data point is collected that does not agree with the model, then a transition test is performed. Transition points contain more information regarding the vehicle location because they must satisfy the following three inequalities.

$$\begin{aligned} |d[k-1] - \mathbf{A}_m \omega[k-1]| &\leq \varepsilon_m \\ |d[k] - \mathbf{A}_m \omega[k]| &> \varepsilon_m \\ |d[k] - \mathbf{A}_{m+1} \omega[k]| &\leq \varepsilon_{m+1} \end{aligned} \tag{4.3}$$

The first inequality shows that the vehicle was in segment n at the data point immediately preceding point d. The next inequality shows that the vehicle is no longer in segment n, and the third inequality shows that the vehicle is now in segment $n+1$. The latter two inequalities have domains that only overlap at the transition point and therefore provide strong evidence about the location of the vehicle.

When segmenting the model map, the error associated with the first model is also recorded. This provides an additional point of reference when determining the validity of a transition. Thus transition points contain information about the vehicle's location that is significantly larger than ordinary points. Segmenting a map with as many transition points as possible improves the localization speed and increases the robustness of the map to noise.

4.3.3 Tracking a Vehicle using the Extracted Models

Once the vehicle's starting point has been identified, the set of models at the bottom of the model tree, or a switched linear system, can be used to track the vehicle's progress. This is a critical function of the algorithm because it provides a method of detecting changes in direction, sudden maneuvers, etc. Tracking is performed in a similar manner to localization, but only a single feasible segment is tested at a time. While the vehicle is located in a segment, its location can be verified using the simpler test shown in equation (4.2), and its position on the map can be updated using odometry readings. This process continues until a data point is found which does not agree with the model.

The data point that does not agree with the current model is assumed to be a transition point and is evaluated using equation (4.3). Additionally, the vehicle location is verified by comparing the distance since the last transition point to the previous segment length. If the data satisfies both checks, then the models are iterated and the process is repeated. Otherwise, the initial localization is assumed to be erroneous. In this case, the vehicle localization loop is repeated and a new location is selected.

In general, once a correct transition point is located, only large changes in the road surface or unexpected maneuvers will lead to errors in the tracking phase of the algorithm. For example, if the driver of the vehicle executes a sudden braking or accelerating maneuver, large oscillations will be introduced in the data. This will lead to a rapid elimination of the feasible models. When this occurs, the tracking code will restart all levels of the model tree that have no feasible models. Practically this means that re-localization is occurring in a neighborhood of the last known vehicle location, ignoring the data produced during the unexpected maneuver.

4.4 Practical Considerations

4.4.1 Advantages of Linear Model Based Localization

The dynamical models extracted from the vehicle pitch data represent a mapping of the underlying road surface. It is not possible to claim that these models are uniquely correlated to a location on a global scale; indeed, there are many situations where road profiles in one location are quite similar to road profiles in a very different location. However, as discussed in the previous section, given a sufficient travel distance, a unique sequence of models can be found for localization; in other words, two roads may be similar to each other for a small model segment, but they will not be similar to each other over many segments in a series.

Using dynamical models offers several advantages. First, given a sufficiently unique map and a noise-free environment the vehicle can be accurately localized for every experiment. While this localization may be to within a neighborhood of the correct position, it is not a probabilistic neighborhood, but rather a deterministic range determined by collection sensor characteristics. Second, in contrast to previous approaches to localization, subsequence identification, and pattern recognition, the approach presented here does not require a metric for matching. Instead the modeling output error is compared to the modeling error bound, equation (4.2), which is determined for each model during the map creation. This simultaneously reduces the need for a metric function, calculating the metric function, and empirically determining a threshold for the metric output. Third, following above, the algorithm is most computationally expensive during the extraction of the model map. This is advantageous because the largest computational constraints are in-vehicle, where computational power is constrained by cost and energy efficiency. Lastly, the linear treatment of data allows for the implementation of simpler noise mitigation strategies. This again reduces the computational power necessary during vehicle travel.

4.4.2 Computational Burden

Because computational power of in-vehicle computers limits the usefulness of most self-contained localization strategies, it is critical to design algorithms that are efficient in their online execution [6]. In particular, the research community in mobile robotics has invested significant efforts in the real-time implementations of the SLAM algorithms [181, 182, 183, 136]. In this section, the computational cost of the presented algorithm is compared to the particle filtering approach in [10]. This comparison is particularly relevant because the particle filtering approach is similar to the prevalent approach used to solve the SLAM problem, and second, the approach in [10] utilized the same database as the work presented here.

The measure used to assess computation cost in [10] is the number of floating point operations (FLOPs) required during the solution of the algorithm. The number of FLOPs required for particle filter convergence is dependent on the number of particles used per mile. For 1000 particles per mile, the particle filter approach required 37009 FLOPs per mile for each iteration during the localization process. Given this number of operations, the in-vehicle computer was only able to scan several miles of the road map during localization.

For the model-based localization approach in this thesis, an adequate vehicle map covering several miles would require many thousands of models. This is because for a map to be considered adequate, it must cover a region large enough to span the uncertainty in vehicle position. The size of this map would result in a very large initial computational burden.

One way to reduce the initial computational burden is to create a tiered tree structure. At the top level of the structure there are "coarse" models that describe large segments of the map. At the bottom level of the structure there are "fine" models that describe small sections of the map. Fig. 4.2 shows an example of this type of model structure. The models are denoted as $\mathbf{A}_{m,l}$, where l is the level in the model structure, and m is the segment index on the m^{th} level. Each model structure will have L levels and M_L models per level. At each successive model level, the ε-region is contracted so that each segment from the previous level is both bisected and modeled with a tighter error bound. This leads to an increase in the number of segments for each consecutive level.

The model structure extracted for this chapter uses bisection at successive levels, i.e. at each consecutive level, the segments from the previous level are bisected. Practically, bisection is advantageous because eliminating large segments at the top of the model structure eliminates a large number of models at lower levels and reduces the computational cost. Additionally, there is little difference between finding a single transition or multiple transition points at a time. This is because there exist naturally occurring transition points between different models which result from changes in the data. These transition points occur for all segmentation types and consequently result in equivalent overall performance.

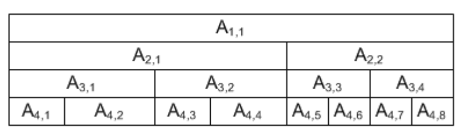

Figure 4.2. An Example Model Structure Designed For Localization.

The localization process discussed in Section 4.3.2 can also be extended to the model structure. In short, the feasible models on each level of the structure are tested sequentially. In this case a model is considered feasible when both the model and its parent have not been invalidated. For example, testing begins at the top level with a single model that describes the map. Then at the second level, two segments are tested. Because the first segment describes the entire map, all segments on level two begin as feasible. Then on the third level only segments that were described by a feasible model in the previous level are tested. This process is iterated to the bottom of the model structure. At each successive level, the models become more precise because the error bound ε becomes tighter and therefore a greater number of segments can be eliminated. The localization procedure is described by the algorithm in table 4.2.

Fig. 4.3 demonstrates the reduction of the computational requirements with respect to distance traveled. In this figure the vertical axis represents the number of floating point operations performed per iteration, and the horizontal axis represents the vehicle's travel distance in meters. The sample point spacing during vehicle travel is 0.5 m, which is a decimation similar to that in [10]. Note that, to process the whole map, the initial number of floating point operations is 36000, but within several meters the majority of the models in the structure have been eliminated and the steady state number of FLOPs per iteration is around 9600 for each incremental map query. The periodic spikes that occur in steady state are transition points that require a larger number of operations to verify.

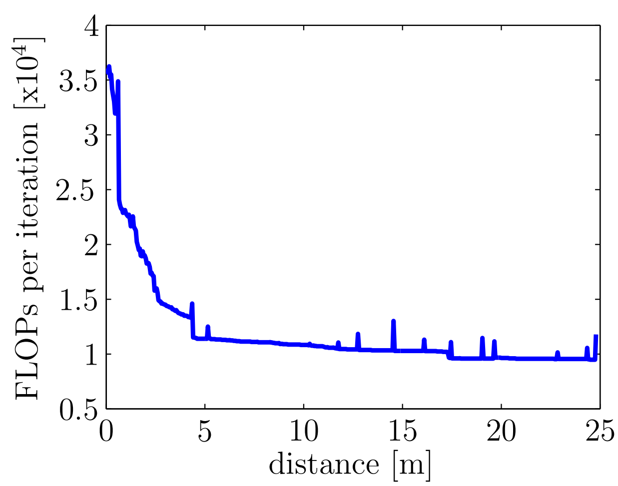

Figure 4.3. An Example FLOPs Count, for Computational Comparisons.

The results presented in this figure are typical for this map but can vary with respect to the starting point of the simulation. As a point of comparison with [10], note that testing was performed on a 6 km (3.73 mile) stretch of road. At the time 6 km was the largest stretch of road that was tractable to store and parse in a vehicle using the PF approach. Comparing this same map, the algorithm presented in this chapter required 9600 FLOPs per mile and per iteration. After convergence, the algorithm requires 2574 FLOPs per mile and per iteration. In this case we note a fourfold reduction in FLOPs. The steady state number of FLOPs required can be further reduced by optimizing the code and using model similarities. Therefore, the size of the map that can be computed with this approach is many times larger than the map that is feasible using a particle filter. It is important to note that the approach presented here is extensible and capable of accommodating any data set size. The main limitations are the in-vehicle computation power and storage capacity, both of which are already within the domain of present desktop computing.

4.4.3 Measurement Noise and Pitch Profile Variance

There are two significant sources of error that affect changes in the road pitch profile that may occur between road traversals: sensor noise and changes in the road pitch profile. The latter is a significant problem because the observed pitch profile is dependent on environmental and mechanical circumstances that surround the vehicle. Even for the same vehicle, different characteristics can alter the observed road profile. These characteristics include the number of passengers, the weight in the trunk, the level of fill in the fuel tank, the state of the system shock absorbers, the amount of wind resistance, the pressure in the tires, and even pot holes. This could be interpreted as a variance in the pitch profile of the road. However, repeated measurements of actual roads show that the road grade profile is invariant or at least can be bounded by a fixed measurement error. Because the goal of this chapter is to demonstrate the approach of compressing data using linear models and then efficiently parsing through this data online, the pitch profile is assumed to be invariant and repeatable with every vehicle run. The remainder of this section discusses IMU noise.

4.4.3.1 IMU Characterization

An IMU contains three gyroscopes that provide an angular rate measurement for each of the 3-dimensional coordinate axes and three accelerometers that output velocity rate information. Although the averaged rate measurements are accurate over time, they are integrated with respect to time to obtain orientation, position and velocity estimates. This integration leads to the accumulation and propagation of small errors.

IMU noise is characterized by the manufacturers in terms of Allan variance of the angular rate output. The primary components of this noise are an angle random walk noise and bias noise. Angle random walk and bias are added to the angular pitch rate before it is integrated to obtain the orientation, position and velocity estimates. Thus even small errors are propagated indefinitely through the integration process.

There is a wide range of commercially available IMU sensors that each exhibit varying noise characteristics. Fig. 4.4 uses the code developed in [146] to demonstrate the noise characteristics for a noisy IMU, ADIS16367; a mid-grade IMU, Crossbow 440; and a low noise IMU, Honeywell HG1700. The top of the figure shows the angle random walk component of the IMU noise and the bottom of the figure shows the bias noise component. In both plots the horizontal axis represents vehicle travel distance in meters.

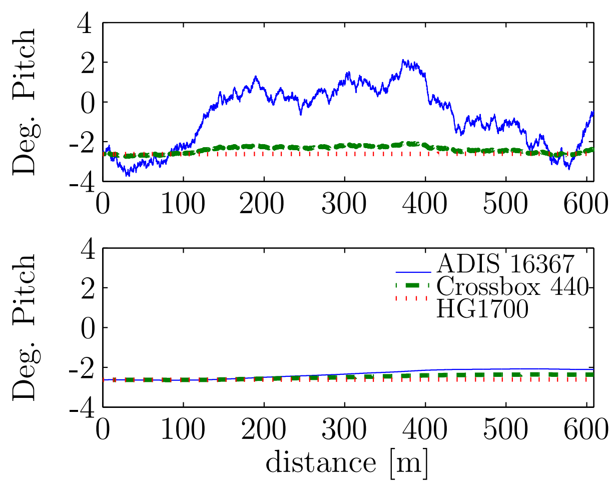

Figure 4.4. Top: Angle random walk noise. Bottom: Bias noise.

Note that the larger component of noise is the angle random walk noise. The angle random walk noise can be described as white noise added to the angle rate measurements of the gyroscopes prior to integration. The standard deviation of the white noise is specified by the manufacturer for a 1Hz sample rate. The specification at 1Hz means that the variance of noise is scaled by the sampling frequency at the output of the IMU. Table 4.3 shows the maximum and average SNRs using the manufacturer specifications and the test data. SNR is calculated using equation (4.4) [184] where σ_m is standard deviation of the map data, σ_n is the standard deviation of white noise provided by the IMU manufacturer, and f_s is the sampling rate at the IMU output.

$$SNR = 20log_{10}\left(\frac{\sigma_m}{\sqrt{f_s}\sigma_n}\right) \tag{4.4}$$

There is an important observation that can be made from the preceding discussion; as the IMU sampling frequency increases, the average SNR monotonically decreases. Because the test map data has been re-sampled to place equal spacing between samples, the effect of noise on vehicle localization can be reformulated as a question of localization resolution. For example, a 100 Hz sample rate corresponds to a 0.05 m map decimation, while a 1 Hz sample rate corresponds to 5 m between samples. According to equation (4.4), reducing the sample rate from 100Hz to 1Hz improves the SNR by a factor of 2. Thus the sampling frequency can be decreased to improve SNR and algorithm performance. This can be done for low-cost sensors provided a coarse map resolution is acceptable.

4.4.3.2 Bias Noise

The first noise component to be addressed is bias noise. The bottom of Fig. 4.4 shows that bias is relatively constant in neighboring data points. Such a low-frequency or DC signal can be rejected through the use of the differences between neighboring data points during map building. Therefore, assuming that bias remains constant in neighboring points, a map of differences can be generated, where each point is the difference between two neighboring pitch values. This is illustrated by the following equation,

$$\Delta d[k] = d[k] + \beta[k] - d[k+1] - \beta[k+1] \tag{4.5}$$

where $d[k]$ and $d[k+1]$ are adjacent pitch values with bias $\beta[k]$ and $\beta[k+1]$, respectively. Because bias is relatively constant, $\beta[k] \approx \beta[k+1]$, the variable $\Delta d[\cdot]$ is insensitive to bias noise. Hence, $\Delta d[\cdot]$ can be used during the extraction of the road map models and structure and during the online localization to mitigate the effects of bias noise. The two data sets, pitch data and angular pitch rate data, are illustrated in Fig. 4.5. To provide context for the data, the approximate route along which the data was collected is included in Fig. 4.6. This figure demonstrates that the test set of data included both measurements from highways and arterial roads.

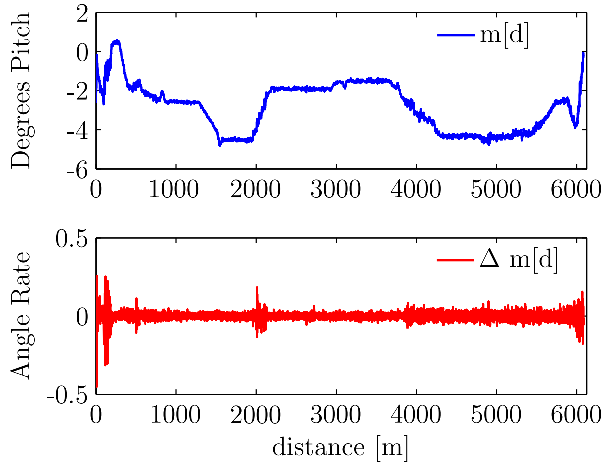

Figure 4.5. Top: Pitch Values vs. Difference Map Bottom: Difference Map

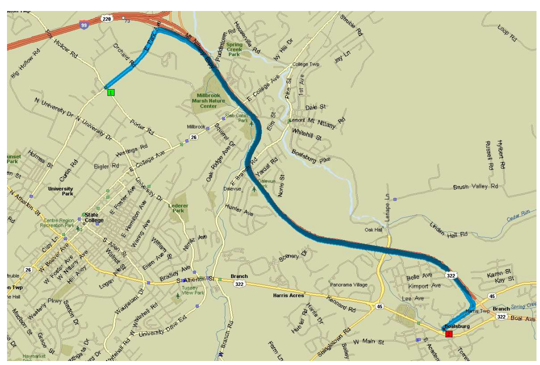

Figure 4.6. Approximate Vehicle Data Collection Route in State College, PA, USA

In Figure 4.5, the horizontal axis represents distance of travel in meters. The top half of the plot shows the pitch data profile used in these experiments. The bottom half of the plot shows the difference map profile of the data. These plots show clearly that any bias integrated into the pitch signal is greatly reduced in the difference map and is very small when compared to the terrain signal size. In particular, when using the bias characteristics of the mid-grade sensor, the residual bias terms were found to be on average four orders of magnitude smaller than the terrain signal size.

4.4.3.3 Angle Random Walk Noise

The second mitigation strategy addresses angle random walk noise. This noise is modeled as white noise added to the angular rate output of the IMU. Otherwise stated, angle random walk noise adds small perturbations to each acquired data point. These perturbations cannot be eliminated by subtraction like the bias noise above. An alternative mitigation strategy is to introduce a tolerance for each data point.

Assume that the perturbations can be bounded by some η_B. Let each data point have a perturbation $|\eta[k]| \leq \eta_B$. Then a model is said to be feasible or agree with the data, contained in vector $\omega[k]$, if a set of constants $\bar{\eta} = [\eta[k], ...\eta[k+N-1]]$ can be found such that,

$$|\Delta d[k] - \mathbf{A}(\omega[k] + \bar{\eta})| \leq \varepsilon. \tag{4.6}$$

The choice of the bound η_B is simplified because we know from the manufacturer that the added noise can be modeled as white Gaussian noise with a standard deviation σ specified in the IMU datasheet at a 1Hz sampling rate. Thus the bound η_B is set to two standard deviations of the additive white Gaussian noise for the particular sensor and sampling frequency.

One important observation is that because pitch is integrated from angular pitch rate, the perturbations at each pitch data point are not independent. Testing a large horizon of data points simultaneously allows for the greatest ability to detect and mitigate the correlation between the points. Practically, this scales the computations of the model validation step and it is therefore not feasible to test more than a sequential few points at a time.

4.4.4 Simulation Setup

When setting up the numerical experiments, the two critical parameters are model order, N, and error bound, ε. Unfortunately, there do not exist optimal values for either parameter. Instead these parameters can be varied with respect to one another to create different model structures with similar localization properties. For the numerical experiments in this chapter, the model order was held constant at five, and the error bound was allowed to vary for each model.

The initial value for the error bound is 0.2556 degrees of pitch. This is the smallest value of ε such that at the top of the model structure, a five-coefficient model can be used to describe the entire data segment. For each successive level, the error bound was contracted by a factor of 0.85.

4.5 Numerical Results

4.5.1 Localization using Noise-free Data

In the ideal case, the traveling vehicle records noise-free pitch measurements from an invariant pitch profile. In this ideal case, given all available data, the algorithm presented will always converge to the correct location. Because of this, the presentation in this section is focused on illustrating the underlying method and demonstrating that the algorithm is coded correctly, rather than analyzing the localization algorithms convergence properties in the presence of noise.

Fig. 4.7 demonstrates one approach to illustrating the algorithm's convergence. This figure shows the convergence of 100 trials using random start points in a noise-free environment. The horizontal axis in the plot shows the distance traveled by the vehicle in meters. The vertical axis shows the number of trials that have converged. Note here that more than 90% of the trials have converged by 25 m of travel distance. All trials have converged within 60 m of travel, which corresponds to the length of the largest segment on the lowest level of the model structure. Thus for the map used in this work, in a noise-free case, the maximum travel distance to convergence can be bounded by the length of this segment.

The variation in convergence distance can also be explained using transition points. Because these points provide a greater amount of information for convergence, the intersection of multiple transition points on the vehicle path leads to shorter convergence distances. This can be observed on the plot where a majority of segments are eliminated within 25 m. The paths that require longer convergence distances lie in more information sparse regions of the map and need a longer time to acquire the information-rich points needed to localize the vehicle.

The figure underscores two of the fundamental advantages of the developed

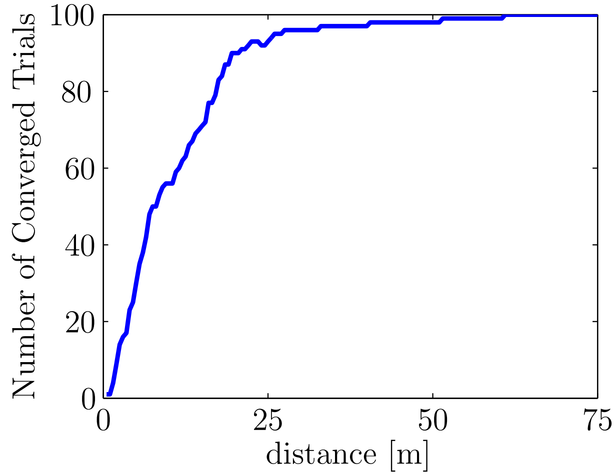

Figure 4.7. Number of Converged Trials per Iteration Number for 100 Noise Free Localizaiton Trials in Vehicle Data.

algorithm. The first is that there exists an upper bound on the convergence time of localization. The second is that the convergence distance can be predicted from the observation of models along the vehicle route.

The processes of segment elimination and transition point identification are illustrated by Figs. 4.8 and 4.9. In these figures the horizontal axes show the distance traveled from the map origin. The top plots show the model output plotted against the reference map data, with the error bounds, ε plotted in red dotted lines. The vertical axis in these plots represents the angle rate. In the bottom plots the model error, in degrees, is plotted against the distance traveled in meters. In this case, the vertical axis represents the model errors.

Suppose that the vehicle begins traveling at 30 m on the map. Then comparing the output of model 1 in Fig. 4.8 with its error bound shows that the segment is a feasible set of vehicle locations. While the vehicle is traveling, all model output errors remain below the modeling bound, ε. However, when the vehicle reaches the end of the segment at 110 m, the output error is significantly higher than the bound ε. This output error is compared to the known transition point error, which was recorded during model identification, for Model 1, and the input data is also compared to Model 2. The mathematical description of a transition point is described in equation (4.3). If all three equations are simultaneously valid and the error matches the known transition point error, then the point at 110 m is identified as a transition point, and Model 2 becomes the feasible vehicle location

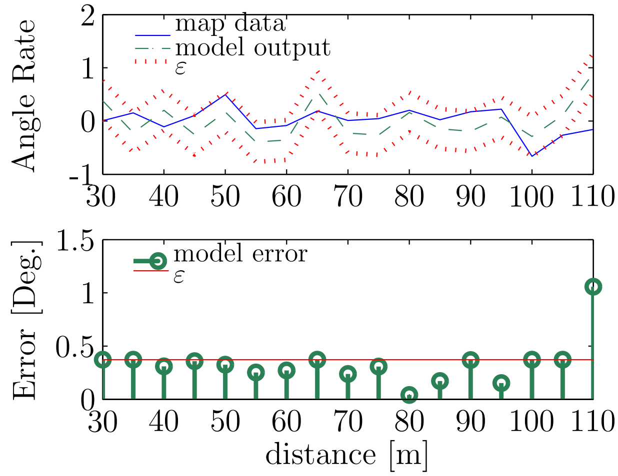

Figure 4.8. Top: Map data vs. Model 1 output, Bottom: Model Error

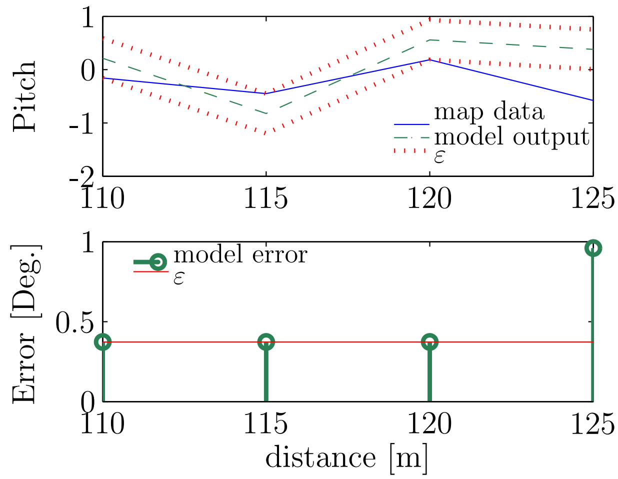

Figure 4.9. Top: Map data vs. Model 2 output, Bottom: Model Error

model.

The comparison of errors is performed for all feasible segments each time a new data point is collected. Because only feasible segments are evaluated, and because only segments whose parents are feasible are re-activated, the number of feasible segments monotonically decreases as the number of iterations rises. Thus the number of feasible segments in Fig. 4.7 converges to one as the number of iterations increases.

4.5.2 Localization using Noisy Data

To test the algorithm in the presence of noise, the map data used to create the reference map is corrupted with the noise characteristics corresponding to the Gaussian random walk characteristics of the Crossbow 440 IMU. The corrupted data is then used for the localization experiments in this section. The map data is plotted along with the road pitch profile in Fig. 4.5, and the approximate real-world map location is shown in Fig. 4.6. At a 1 Hz sampling rate, or a 5 m map decimation, the Crossbow sensor has noise characteristics that are an order of magnitude higher than the collection sensor but sufficiently low to allow for accurate localization. The algorithm performance is evaluated over a 2 km travel distance from a random start point that is uniformly chosen.

Fig. 4.10 shows a histogram of the vehicle travel distance required for algorithm convergence in over 1100 trials with a random start point. As before, the convergence to a single path corresponds to the first estimate of vehicle location. The horizontal axis in the figure represents the vehicle travel in meters, and the vertical axis represents the number of trials. The mean convergence time for this set of trials is 172 m. Furthermore the maximum convergence distance is about 280 m of travel.

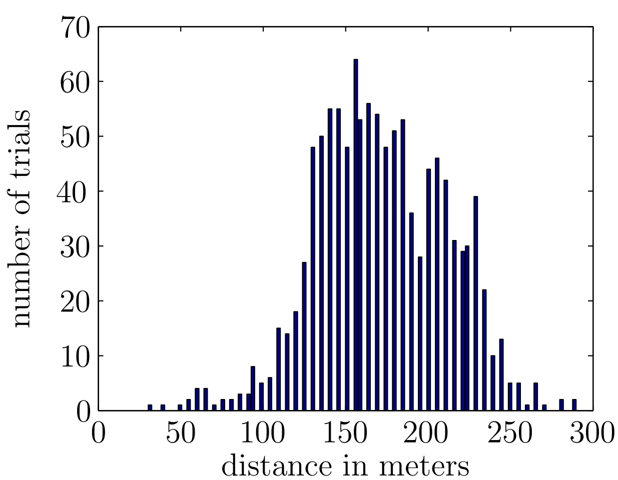

Figure 4.10. Distance traveled until 1 feasible path using Crossbow 440 IMU

For trials in the presence of noise, the plot of localization errors [m] vs. the trial start points [m] in Fig. 4.11 helps to illustrate the algorithm's performance across the map. For example, note that the localization distance is different in the beginning of the map when compared to the end of the map. This difference comes from the physical nature of the roads from which the data was collected. In this case, the road vehicle began traveling on a secondary road and merged onto a highway. Thus the beginning half of the map contains data whose variance is high, and the latter portion of the map contains highway data with low variance. In the presence of high variance data, models have larger modeling error bounds and are eliminated at a slower rate. This fact is illustrated in the convergence error, which is highest in the beginning of the map. Examining the convergence errors more closely reveals that when errors are measured in terms of map decimations (5 m), all errors fall within a few decimations.

The accuracy achieved in these simulations is comparable to the findings of Dean [10], who found that the limiting factor in his particle filtering approach was the map decimation. The results also compare favorably to the accuracy found in recent map-based localization work [175, 176] which had localization accuracies of about 1 m and 15 m, respectively. While [175, 176] do not use IMU data for localization, the research presented in these papers is map-based localization research using real-world noisy sensors. These references show that the work presented here is at least as accurate as the state of the art in localization, with the additional benefits discussed throughout the chapter. A strength of the approach presented here is that localization is a function of the map decimation, as pointed out in section 4.4.3.1. In particular, map decimation is proportional to the size of the observed sensor noise, and thus choosing higher fidelity sensors will allow finer resolutions and correspondingly smaller localization errors.

Overall, the algorithm performs well using the noise characteristics of the Crossbow 440 sensor. It is possible to address the source of error and decrease convergence distance by adjusting the mechanism by which the likely vehicle path is selected among the remaining paths. This promising area of research is left to future work.

4.6 Discussion of Algorithm Limitations

The preceding sections tested the terrain-based localization algorithm for an invariant pitch profile with additive sensor noise. As can be seen in Fig. 4.11, the

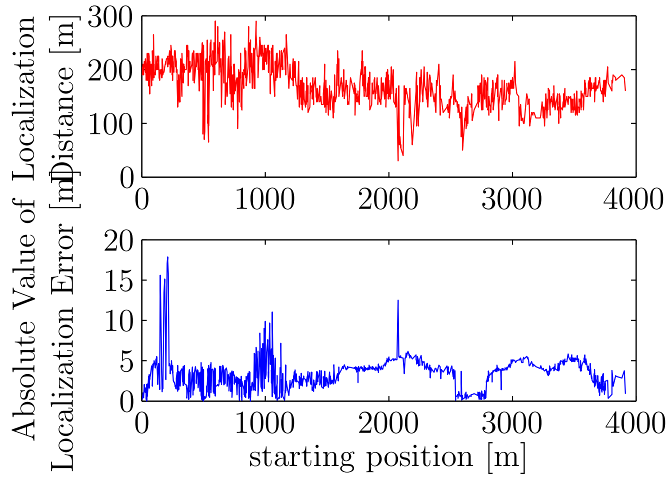

Figure 4.11. Convergence Distance and Error vs. Start point

localization results using this algorithm are good for the mid-range IMU sensor. On average the error observed in the location estimates is on the order of the map decimation.

There are two principle sources of error in the localization process: survival of too many paths in the presence of noise and erroneous segment elimination. The first is a natural consequence of using terrain data and a bounded localization distance in the demonstration of this algorithm. Terrain data is unique on a local scale but not on a global scale. Thus when the localization algorithm is working with an area of the map that has low SNR and does not display strong uniqueness in features, multiple models survive. For example localization might be difficult in the Great Plains of the United States where the terrain is largely flat resulting in fewer road features. Conversely, localization might be more rapid in the Rocky Mountains (United States) where the topography is quickly changing and offering a large number of unique terrain patterns.

Here it is important to note that reference maps are made up of patterns that are used for localization. These include both the geographical effects and the road construction effects (e.g. the sequence of concrete or asphalt pours). Thus, even in flat terrains, there may still remain a surprising number of localization features due to location specific construction of the road.

This chapter demonstrated a series of localization experiments for a set road distance. Making the road distance finite was chosen to make the experiments tractable. If multiple models survived, a heuristic based on the number of detected transitions was invoked to choose the correct path. This is not an optimal choice since noise can also trigger false transitions. In order to eliminate heuristic errors, the solution is to extend the run of each experiment until it is localized. However, the results above show good performance for a fixed localization distance.

Conversely, the presence of noise can also lead to the erroneous elimination of segments. For instance, the angle random walk mitigation approach presented in this chapter bounds the possible noise coefficients to two times the standard deviation of the angle random walk noise. This means that in approximately 5% of the cases it is expected that the noise will exceed the limitation and the correct segment will be eliminated. As travel distance is increased indefinitely, these types of eliminations are corrected by segments in higher levels of the tree with larger error tolerances. However, for a fixed distance, the end result may appear erroneous because of one of these eliminations. In this case, the algorithm can be re-initialized and restarted.

4.7 Conclusions and Future Development

This chapter presented two linear model based algorithms specifically designed for localization. The first algorithm is used to compress a one dimensional vector of data and to represent this data as a tree of models with increasingly tighter modeling error bounds. The second algorithm is a parsing algorithm that can quickly locate an incoming stream of data in the extracted model tree. Both algorithms were developed and demonstrated for the application of vehicle localization. Because these algorithms and the linear model representation were specifically developed for in-sequence localization, the work here greatly outperforms the work reviewed in the previous chapter.

Thus with the conclusion of this chapter a representation has been designed that address all but one of the required characteristics - robustness to noise. In the subsequent chapters the models will be tuned to accommodate the expected noise in the process. In essence, this work will show, that given some further information about the process, it is possible to improve the linear modeling approach to data representation by choosing models that will be more robust in the presence of noise. Lastly, because robustness must also include redundancy both for increased accuracy and reliability, the next chapters will also extend the development of this representation into multiple dimensions.

Vehicle Localization
Initialization
Collect $N+1$ data points:
set loop index: k $= N+1$
initialize data vector: $\omega[\cdot] = d[1:N]$
Begin Localization Loop
WHILE localization flag == 0
FOR l = 1:L
FOR m = 1:$M_L - 1$
IF the parent segment of$\mathbf{A}_m$ is feasible:
$e_{m,k} = \lvert d[k] - \mathbf{A}_m \omega[\cdot] \rvert$
$e_{m+1,k} = \lvert d[k] - \mathbf{A}_{m+1} \omega[\cdot] \rvert$
IF $e_{m,k} < \varepsilon_m$
set $\mathbf{A}_m$ as feasible
ELSEIF $e_{m,k} > \varepsilon_m$ AND $e_{m+1,k} < \varepsilon_{m+1}$
then the data point m[d] is a transition point
obtain τ_n from the map
ELSE
set $\mathbf{A}_m$ as infeasible
END IF
END IF
END FOR
IF l = L AND num. feasible models = 1
localization flag = 1
END IF
END FOR
Collect an additional data point
iterate the index: k = k + 1
iterate the data vector:$\omega[\cdot] = d[k-N:k-1]$
END WHILE
Following a Detected Transition Point
FOR l = 1:L
FOR m = 1:M_L
IF τ_n is inside the segment
Set the segment as feasible
ELSE
Set the segment as infeasible
END IF
END FOR
END FOR
Exit to Tracking Loop

Table 4.2. Vehicle Localization Algorithm for Model Structures

Sensor	f_s [Hz]	Average SNR [dB]
ADIS16367	1	-6
Crossbow 440	1	12
Honeywell HG1700	1	34

Table 4.3. Vehicle Data SNR at 1Hz Given a Sensor Choice.

Chapter 5

Robust ARX model-based data structures for In-Sequence Localization using 1-Dimensional Time Series

Building on the work in the previous chapter, this chapter focuses on tuning the linear model data representations for robustness in the presence of noise. The disturbances considered here are both deterministic and stochastic. Deterministic noise is often observed due to interference between devices [185] or vibrations in the recording sensors [186]. Stochastic noise is typically observed in any practical data collection device and is an important component in algorithm performance. The consideration of both types of noise is critical in data representation [187].

The deterministic noise considered in this chapter is sinusoidal noise. This noise, which occurs in IMU data [186] and is of interest in many other applications including speech processing [185], is particularly destructive to data representations because strong cyclic noise components can obscure the data points about which representations are determined. In this chapter, the internal model principle is used to automatically remove sinusoidal components of a known frequency during the in-sequence localization process.

The stochastic noise considered in this chapter is white Gaussian noise. This noise can be equally destructive as shown by the results in chapter 3. In particular, in IMU data, the effects of the noise are accentuated when the data is integrated to obtain quantities such as position and velocity [146].

In addition to introducing the idea of robust tuning of the models, this chapter also improves the localization process by making it adaptive such that localization occurs after a user defined accuracy is achieved. Moving to an adaptive in-sequence localization process based on robust data representations demonstrates the effectiveness of the representations for in-sequence localization. In addition, when viewing the results of localization in parallel with the representation map, the adaptive approach to in-sequence localization reveals the strengths and weaknesses of the data sensor, which leads to a logical method of evaluating the addition of sensors with respect to the amount of information they contribute to the localization process.

Lastly, this chapter seeks to extend the domain of the problem of in-sequence localization. In addition to experiments on vehicle data, new numerical experiments are presented using random data. These experiments show that any data set can be put into the algorithms developed in this thesis. In addition US inflation data is used to demonstrate the practical significance of transition points. In fact, work in this chapter shows that when inflation data is segmented, the resulting transition points delineate the beginning and end of major US recessions and depressions.

The preliminary work that supported the development of this chapter was presented in the 2013 Penn State College of Engineering Research Symposium [25] where it was awarded the best paper award. In addition, separate papers were presented at the 2013 IEEE Conference on Decision and Control [23] and the 2014 American Control Conference [24]. A full manuscript detailing the work of this chapter is currently in review with IEEE Transactions on Knowledge and Data Engineering [167].

5.1 Introduction

The objective of this chapter is to further the design of one dimensional data representation that are robust to noise for in-sequence localization. By designing the localization map (which is actually a data representation structure) in a way that reduces the effects of noise, the online matching process is primed to be optimized to further reduce the computational load, enabling larger data structures to be parsed. Here we consider additive sensor noise that is both deterministic and stochastic and describes commonly observed noise in real-world data. The next subsection, briefly overviews the history of sensor noise in the data representation literature. The main drawback of traditional dimension reduction approaches is highlighted with a brief set of simulations.

5.1.1 The Effect of Noise on Data Representations in Literature

Recall from the discussion in Chapter 3 that in the presence of noise the traditional dimension reducing representations are not adequate (see chapter 3 or [27, 28, 29, 30, 31, 32]) for the case of in-sequence localization for three reasons. First, traditional methods require windowing the data which introduces errors if data acquisition begins in the middle of a window. Second, few of these representations have an inherent mechanism for mitigating the effects of noise. And third, these representations do not preserve the ordering of the data, which is a key attribute of the data that can be exploited for in-sequence localization.

As a brief example, consider Fig. 5.1 where the top plots show random data encoded into three popular representations: the data mean ([29]), the data slope ([31]), and DFT coefficients [47]. The bottom plots show the data encoded into the same representations, but this time the encoding occurred after the addition of noise. The experiments were performed on a random data set generated in MATLAB, and the data is corrupted with approximated 12 dB of Gaussian noise and 12 dB of linear noise for consistency with previously published work by the authors. Note that the addition of noise significantly alters the resulting representations as demonstrated numerically in the caption of the figure.

Some preliminary work [23, 24], has shown that it is possible to design data representations that are inherently robust to noise. The key to finding these representations is selecting points in the data that are inherently robust to noise, i.e. they display behavior that is significant enough that it cannot be obscured by the

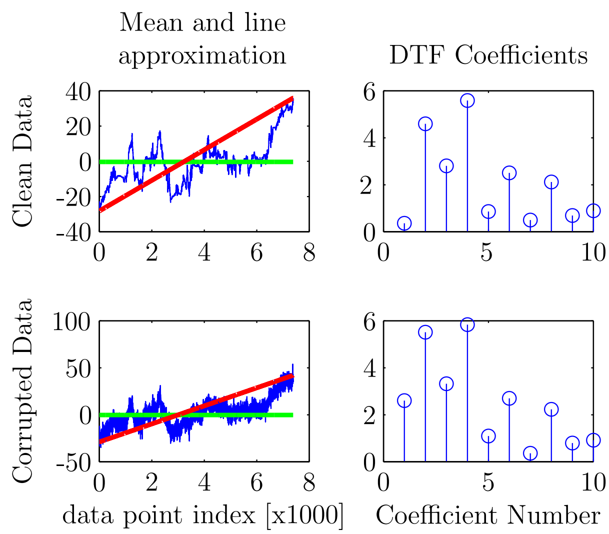

Figure 5.1. Clean Data: mean = -0.36, slope = 64.17. Corrupt Data: mean = 2.60, slope = 70.97

additive sensor noise. In this chapter, these points are chosen to reflect significant changes in the data dynamics, where quantifying significant depends on the characteristics of the observed noise.

5.2 Overview of Dynamical Model-based Data Structures

5.2.1 Using Dynamical Models to Represent Data

Prior to describing the identification of robust data representations, this section provides a brief overview of the dynamical modeling approach used in the last chapter [20, 23]. Then section 5.3 will discuss the contributions of other authors in the field of subsequence matching, and an in-depth discussion of the approach taken in this chapter begins in section 5.4.

Dynamical models have inherently advantageous properties as data representations. These models are capable of preserving data information, modeling data modes, and detecting changes in data dynamics to provide natural transition points. To describe a data set using dynamical models, $\mathbf{D}$ is first segmented into non-overlapping intervals. Each interval, m, is represented using an autoregressive linear model of N^{th} order, A $= [c_{m,1}, \ldots, c_{m,N}]$, with a modeling error bound ε_m. [1] The beginning and end data points of an interval m are called ***transition points*** and denoted by τ_{m-1} and τ_m, respectively. Transition points are points in the data at which the underlying data dynamics are changing. This change in dynamics at τ_m is denoted by a modeling error that exceeds the modeling error bound of the preceding data model. Maintaining the ordered nature of **D**, the ordered set of models describing sub-intervals is shown in equation (5.1).

$$\begin{aligned} d_k &= c_{m,1}d_{k-1} + ... + c_{m,N}d_{k-N} + e_k, \ \forall k \in [\tau_m - 1, \tau_m) \\ d_k &= [c_{m,1}...c_{m,N}]A_m^\top + e_k, \qquad\qquad \forall k \in [\tau_m - 1, \tau_m) \\ &\qquad |e_k| \leq \varepsilon_1 \qquad for\ m = 1, 2, \ldots \end{aligned} \tag{5.1}$$

To create a reference data structure for in-sequence localization, the data is sequentially segmented into smaller intervals whose models describe the data more accurately. Each sequential segmentation is stored into a model tree that aids in the online localization process. For example, a data set is first segmented into two segments each with a corresponding modeling error bound ε_m. The segments are recorded into the first level of the model tree. At the second level of the model tree, the segments from the first level are broken into two more segments each with a smaller bound ε_m. This process is iterated until the model error bound approaches the size of the expected noise in the data. The modeling structure is depicted in Fig. 5.2.

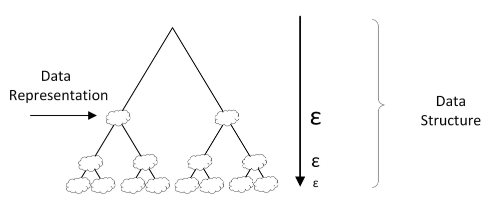

Figure 5.2. A Depiction of Model Structures for In-Sequence Localization.

Localization is the process of sequentially evaluating model agreement with the new data and eliminating models that disagree. Mathematically the process of model evaluation is the determination of the model prediction error and comparing

[1]These models are found by solving a linear feasibility problem. For this reason the models in previous work are not necessarily unique.

it to the modeling error bound, as shown in equation (5.2).

$$\begin{aligned} |d_k - c_{m,1}d_{k-1} + ... + c_{m,N}d_{k-N}| &\gtrless \varepsilon_m \\ |e_k| &\gtrless \varepsilon_m. \end{aligned} \tag{5.2}$$

When evaluating models, errors that are smaller than ε_m indicate that this particular interval is a candidate for the data sequence location. Errors that are larger than ε_m indicate this is not a possible location of the data sequence. When a transition point is reached in the data, three distinct inequalities are satisfied: the m^{th} model was in agreement at the $k-1^{st}$ instant, the m^{th} model disagrees with the data at the k^{th} instant, and the $m+1^{st}$ model agrees with the data at the k^{th} instant.

$$\begin{aligned} |d_{\tau_1-1} - (c_{m,1}d_{\tau_1-2} + ... + c_{m,N}d_{\tau_1-1-N})| &\leq \varepsilon_m \\ |d_{\tau_1} - (c_{m,1}d_{\tau_1-1} + ... + c_{m,N}d_{\tau_1-N})| &> \varepsilon_m \\ |d_{\tau_1} - (c_{m+1,1}d_{\tau_1-1} + ... + c_{m+1,N}d_{\tau_1-N})| &\leq \varepsilon_{m+1} \end{aligned} \tag{5.3}$$

From these equations it becomes clear that transition points contain significantly more matching information than points in the interior of data intervals. Thus while the majority of data points provide a range of possible locations, transition points provide an exact location, or a sparse list of candidate locations.

Objective: Given the importance of transition points, the objective here is to maximize the robustness of the transition points in the presence of sensor noise.

Making transition points more robust will increase the overall robustness of the approach and shorten the necessary data sequence length for matching. A full development of the proposed approach can be found in section 5.4.

5.3 Background

In presenting the literature review, the section follows the logical order of data representation and structuring. Section 5.3.1 shows methods of windowing the data. Then section 5.3.2 discusses representations of the windowed data. Finally, a brief overview of data representations that explicitly discuss noise is provided in section 5.3.3.

5.3.1 Transition Points

The most critical component in a data representation creation algorithm is segmentation - i.e. choosing the points at which to divide a set of data into intervals. These points are referred to as transition points because of the model transitions that these points represent, in other literature these same points are sometimes called change points. Change points were first defined in [33] as the points at which the behavior of the time series changes, a problem also examined by the statistics community. The general approach to solving the change point identification problem is to identify a set of break points through the identification of a set of models about the point [15]. The models are identified by minimizing some loss function specified by the author, and a change point consists of a point about which the identified models have significant dynamical differences. Popular choices for the loss function include the mean square error, least squares fit, and maximum likelihood estimation.

Transition points are also frequently called events, and segment bisection is commonly adopted during the segmentation process [34]. The neural networks literature contains an example of a similar idea where the switching sequence of a subprocess is identified using nonlinear gated experts, a statistical physics tool [35].

Using optimization approaches, Ozay et. al. [188] developed the linear programming-based break point identification algorithm that was the basis for the initial work in Chapter 4 [20]. Using dynamical programming, Duncan and Bryant [36] created an algorithm to find the number of possible data intervals, the model order in each interval, and the location of the intervals. At its core, this was a least-squares fit algorithm that identified models of the data. The work by Ozay et. al. and Duncan and Bryant is a step forward from previously published work that detected changes in dynamical systems [37] based on the parameterization of linear systems in [38].

Lastly, concluding this subsection we mention several sliding window approaches that compete with significant event detection approaches discussed above. To begin, Chu [39] presented a sliding window approach to change point localization and segmentation. This was followed by Fancourt and Principe [40] with a piecewise segmentation and identification procedure. The latter approach maps similar segments in a time series as neighbors in a neighborhood map. More recently Keogh et. al. [41] proposed a bottom up sliding window method termed SWAB.

5.3.2 Data Mining Features

Following the segmentation of the data, each window is converted to a feature or a symbol. Of particular relevance is the paper by Shatkay and Zdonik [27] where linear regressions are used for dimension reduction in large data sequences. Because the goal of this work was simplicity and because the original time series was kept for in-depth matching, the authors choose to represent trends in the data as lines defined from one extrema point of the data to the next. This representation is referred to as the piecewise linear representation (PLR). This paper also contributes to the idea of abstract representations by proposing data representations using only the extrema points of the time series.

Following the approach established by [27], many authors have proposed variants of the first order representations. Prominent examples include the piecewise aggregate approximation (PAA) [28, 29] which represents data using the mean value of a segment. The PAA algorithm stipulates that segments must be of fixed and equal length. In some data sets this is a significant constraint, which prompted work by Keogh et. al. [30] to update the PAA algorithm allowing variable segment lengths. The new approach was termed the adaptive piecewise constant approximation algorithm (APCA). In addition several authors build on the piecewise linear representation creating similarity search algorithms [31] and clusering algorithms [32].

Lastly, the new representations are stored using application-specific structures. Two principle types of structures are used. The first are classification schemes [15] that require the comparison between classes and new data. The second are data trees [189] that allow rapid elimination of possibilities.

The body of research in data representations is extensive. However, several critical review papers [13, 14] have demonstrated that large issues exist with the current set of approaches. The published representations and matching approaches were shown to perform well for small sets of data but their computational performance was shown to rapidly deteriorate as the underlying data size increases. Thus new approaches are necessary to advance the field of data knowledge.

5.3.3 Robust Features

Few papers explicitly discuss or make mention of data noise as a consideration. One possible reason for this lack of literature is that most data mining is performed with the goal of finding "similar," not exact features. This differs in the application of some data representations, for example those used in vehicle localization where an exact identification is needed.

The most common representation, and one which has been discussed in the presence of noise, is PLR [42, 43]. In particular the work by Jia et. al. [43] contains a good review of PLR algorithms and then aims to develop a new approach that improves PLR's robustness to noise. Here the authors note that the need for a user-specified number of segments is a major weakness. By reformulating the problem in terms of error bounds, the authors automatically choose the number of segments in PLR on a data set given an error bound.

A separate subset of papers by Fink et. al. [44, 45] also focuses on the problem of identifying noise robust points. The difference here is that the papers focus on the identification of maxima and minima in the data, which are inherently more robust. The paper by Fink and Pratt [44] builds on this idea by identifying patterns in the reduced dimension (extrema point) data set.

Building on the idea in [44], the work by Vemulapalli et. al. [12] proposes an optimal filter that reduces the effect of noise prior to identifying patterns. The patterns in this work are combinations of robust minima and maxima in the data. Combining filtering and robust pattern identification improves on the classical approach of smoothing the data before obtaining representations. Furthermore, this approach is verified using real-world vehicle data from the author's lab.

Lastly, a more general approach to robust data representations is laid out by Preng et. al. [46]. The work cites the natural method of location recognition by animals and humans, which use landmarks to identify particular sites in their surroundings. The landmark model is defined to be invariant under smoothing and transformations such as shifting, scaling, and warping. However the applications shown by the authors are limited and leave the development of more extensive applications to readers of the paper.

5.4 Developing Dynamical Model-Based Robust Data Representations

5.4.1 Bounding the effect of noise in context of linear model-based data structures

This chapter builds on the dynamical model-based approach previously used by the authors by incorporating knowledge about the sources of noise to make the data representations more robust. This robustness is introduced both at the representation and structure levels. A depiction of the data structure and representation is shown in Fig. 5.2.

On the structure level, two mitigation strategies are employed. First, the structures are tiered with large data decimations at the top leading to lower decimations at the bottom. To achieve the larger decimations, the data is averaged, which leads to a reduction in the noise power particularly for independent zero mean types of noise. Second, the dynamical models have increasingly tighter modeling error bounds, ε, for lower structure levels. This means that as the level of noise changes, the same data structure can be used at varying resolutions. Small levels of noise can be accommodated by more precise models, which provide greater resolution during localization. Conversely, large levels of noise will only be accommodated by coarser models resulting in more coarse localization results. Practically this means that one data structure can be extracted and used with a variety of sensors, ranging from inexpensive to military grade.

At the representation level, the dynamical models characterized in this chapter emphasize the preservation of transition points in the presence of noise. Here it is assumed that the additive noise is Gaussian with a probability distribution function $\mathcal{N}(0, \sigma_\eta^2)$, as described in the notation section. The corresponding cumulative distribution function of this noise is $\Phi(0, \sigma_\eta^2)$. Then, because the models are linear, at the k^{th} data point, the m^{th} model output and output error are also Gaussian random variables. The distribution of the error can be described as $\mathcal{N}_{e_m}(\mu_{e_m,k}, \sigma_{e_m}^2)$ with a corresponding cumulative distribution $\Phi_{e_m}(\mu_{e_m,k}, \sigma_{e_m}^2)$. At a transition point, one model's segment is ending and another model's segment is beginning. Thus the transition point can be described by two overlapping Gaussian distributions, shown in Fig. 5.3: a distribution for the modeling error of the model whose segment is ending and a distribution for the model whose segment is beginning. Similar to the idea used in radar theory [190], the area below the overlap of these distributions represents the probability of an error at a transition point. Since its inception, this idea has been frequently used in many fields, for example in the field of fault detection Iserman [191] has used the idea to differentiate between fault/no fault conditions.

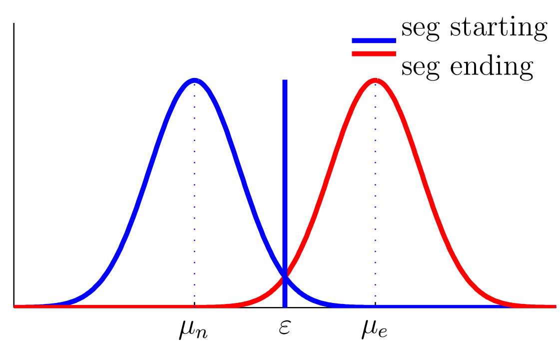

Figure 5.3. The Distribution Overlap Assuming Additive Gaussian Noise.

Using the model coefficients and the additive noise distribution, it is then possible to precisely describe the probability of the m^{th} model agreement ending as $\mathbb{P}(|e_m| \geq \varepsilon)$, and the $m + 1^{st}$ segment beginning as $\mathbb{P}(|e_{m+1}| < \varepsilon)$. Then the optimal choice of a transition point will minimize the probability of an error, i.e. the optimal choice will maximize $\mathbb{P}(|e_m| \geq \varepsilon)$ and $\mathbb{P}(|e_{m+1}| < \varepsilon)$. One simple form of writing this optimization problem is described in equation (5.4).

$$
\begin{aligned}
&minimize \quad \mathbb{P}_{error} \\
&s.t. \qquad \eta_{\hat{k}} \backsim \mathcal{N}(0, \sigma_\eta^2) \\
&\qquad \mathbb{P}_{error} = 2 - \mathbb{P}(|e_m| \geq \varepsilon) - \mathbb{P}(|e_{m+1}| < \varepsilon)
\end{aligned}
\tag{5.4}
$$

In practice to obtain a related convex problem which can be efficiently solved, the optimization problem is written in the chance constrained framework demonstrated by Kataoka [192]. In this framework, the problem is divided into two subproblems denoted by $\mathcal{O}1$ and $\mathcal{O}2$, each of which describes a convex solution domain. For each subproblem the modeling error bound is minimized while the maximum probability of error is bounded by a user-defined value ρ. The modeling error probability is described by $\Phi_{e_m}(\mu_{e_{m,k}}, \sigma^2_{e_m})$, where m denotes the model number, and k denotes the data index in the underlying time series.

There are two resulting formulations used to segment an interval of data $[d_{k_s}, d_{k_e}]$. The first, $\mathcal{O}1$ shown in equation (5.5), finds a model whose segment end error exceeds a positive ε at the transition point τ_m. Then the second formulation, $\mathcal{O}2$ shown in equation (5.6), determines a model whose end error is less than a negative ε. The breaking up of these formulations is necessary to ensure convexity when solving for the models. A more detailed derivation of the formulations is shown in the Appendix.

$$
\begin{aligned}
\mathcal{O}1: \quad & \mathit{minimize} \quad \varepsilon \\
& s.t. \\
& \quad \varepsilon \le \varepsilon_{max} \\
& \quad \mu_{e_{m,k}} > \varepsilon \\
& \quad \sigma_{e_m} \Phi^{-1}_{e_m}(1-\rho) + \mu_{e_{m,k}} \ge \varepsilon \\
& \quad \sigma_{e_{m+1}} \Phi^{-1}_{e_{m+1}}(\frac{\rho}{2} + \frac{1}{2}) + \mu_{e_{m+1},k} \le \varepsilon \\
& \quad |d_k - [d_{k-1}, \ldots, d_{k-N}]A_m^\top| \le \varepsilon \qquad \forall k \in [k_s, \tau_m - 1) \\
& \quad |d_k - [d_{k-1}, \ldots, d_{k-N}]A_{m+1}^\top| \le \varepsilon \quad \forall k \in [\tau_m, k_e]
\end{aligned} \tag{5.5}
$$

$$
\begin{aligned}
\mathcal{O}2: \quad & minimize \quad \varepsilon \\
& s.t. \\
& \quad \varepsilon \leq \varepsilon_{max} \\
& \quad \mu_{e_{m,k}} < -\varepsilon \\
& \quad \sigma_{e_m} \Phi_{e_m}^{-1}(\rho) + \mu_{e_{m,k}} \leq -\varepsilon \\
& \quad \sigma_{e_{m+1}} \Phi_{e_{m+1}}^{-1}(\frac{\rho}{2} + \frac{1}{2}) + \mu_{e_{m+1,k}} \leq \varepsilon \\
& \quad |d_k - [d_{k-1}, \ldots, d_{k-N}]A_m^\top| \leq \varepsilon \qquad \forall k \in [k_s, \tau_m - 1) \\
& \quad |d_k - [d_{k-1}, \ldots, d_{k-N}]A_{m+1}^\top| \leq \varepsilon \quad \forall k \in [\tau_m, k_e]
\end{aligned}
\tag{5.6}
$$

Formulating the model extraction problem using this approach has some significant advantages. First, when compared to previous work by the authors, this approach provides a metric for model extraction. This metric quantifies the significance of the dynamic discontinuities and solidifies the choice of model for each segment. Furthermore, using this metric reduces the number of extracted segments, resulting in model structures that are significantly more compact than previous structures [20, 23]. Lastly, the *a priori* specification of the expected noise level reduces the need for add-on noise mitigation algorithms and transfers noise mitigation to be performed mostly during the data representation extraction and structure creation.

5.4.2 Canceling the Effects of Known Sinusoidal Noise

The approach used in the preceding section can be used for any noise distribution whose inverse can readily be determined. White Gaussian noise is a great example of this type of noise. However, as noted in the introduction there are also many types of noise that are not stochastic. Two types of deterministic noise in IMUs are sinusoidal noise and bias noise [186]. Bias noise, which can be considered a very low frequency sinusoidal noise, can be addressed by working with the data differences instead of the raw data. Indeed the approach we have taken here, is the same as the approach in the previous chapter. However, this approach is not adequate for higher frequencies sinusoidal noise.

One approach to reducing the effects of deterministic sinusoidal noise is to employ the internal model principle. Using the internal model principle shows the flexibility of using ARX models as data representations. To begin suppose that the deterministic sinusoidal noise has a frequency ω_n and is modeled as a complex exponential $e^{-j\omega_n t}$. Now consider the ARX model in equation (5.1). Neglecting the error term, the estimate of the time series can be expressed as,

$$\hat{d}_k = c_{m,1}d_{k-1} + ... + c_{m,N}d_{k-N}. \tag{5.7}$$

Using the Z-transform, the system function of model can be expressed as a polynomial in the delay operator z,

$$\begin{aligned} &(c_{m,1}z^{-1} + ... + c_{m,N}z^{-N}), \\ &(1 - \lambda_1 z^{-1}) \dots (1 - \lambda_N z^{-1}), \end{aligned} \tag{5.8}$$

where $\lambda_1, \dots, \lambda_N$ correspond to the zeros of the system function. When evaluating the frequency response of the system function with respect to the locations of the zeros, the delay operator z is set to $e^{j\omega}$, resulting in the evaluation of the system function equation on the unit circle in the z-plane. This means that for all frequencies ω_i such that $1/\lambda_i = e^{j\omega_i}$, the system will produce on output of zero.

Combining this with the knowledge of the expected noise frequency ω_n, if one of the system zeros is set to $\omega_n = 1/\lambda_n$, the output of the system when excited by this disturbance will be zero. There are two methods of achieving the placement of this zero. The first, is to constrain some of the model coefficient magnitudes in problems $\mathcal{O}1$ and $\mathcal{O}2$. The second, is to convolve a first order polynomial, $1 - \lambda_n z^{-1}$, with the determined model. The second method is preferable because it divides the tuning of the models into two steps, and provides the most flexibility in finding transition points robust to stochastic noise.

$$[1,\ \lambda_n] * [1,\ c_{m,1},\ \dots,\ c_{m,N-1}] \tag{5.9}$$

Using the latter method the updated optimization problems can be written as,

$$\begin{aligned}
\hat{\mathcal{O}}1: \quad & minimize \quad \varepsilon \\
& s.t. \\
& \quad \varepsilon \le \varepsilon_{max} \\
& \quad \mu_{m,k} > \varepsilon \\
& \quad A_m = [1,\ \lambda_n] * [1,\ c_{m,1},\ \dots,\ c_{m,N-1}] \\
& \quad A_{m+1} = [1,\ \lambda_n] * [1,\ c_{m+1,1},\ \dots,\ c_{m+1,N-1}] \\
& \quad \sigma_{e_m} \Phi_{e_m}^{-1}(1-\rho) + \mu_{e_{m,k}} \ge \varepsilon \\
& \quad \sigma_{e_{m+1}} \Phi_{e_{m+1}}^{-1}(\frac{\rho}{2} + \frac{1}{2}) + \mu_{e_{m+1,k}} \le \varepsilon \\
& \quad |d_k - [d_{k-1}, \dots, d_{k-N}] A_m^\top| \le \varepsilon \qquad \forall k \in [k_s, \tau_m - 1) \\
& \quad |d_k - [d_{k-1}, \dots, d_{k-N}] A_{m+1}^\top| \le \varepsilon \quad \forall k \in [\tau_m, k_e]
\end{aligned} \tag{5.10}$$

$$\begin{aligned}
\hat{\mathcal{O}}2: \quad & minimize \quad \varepsilon \\
& s.t. \\
& \quad \varepsilon \le \varepsilon_{max} \\
& \quad \mu_{m,k} < -\varepsilon \\
& \quad A_m = [1,\ \lambda_n] * [1,\ c_{m,1},\ \dots,\ c_{m,N-1}] \\
& \quad A_{m+1} = [1,\ \lambda_n] * [1,\ c_{m+1,1},\ \dots,\ c_{m+1,N-1}] \\
& \quad \sigma_{e_m} \Phi_{e_m}^{-1}(\rho) + \mu_{e_{m,k}} \le -\varepsilon \\
& \quad \sigma_{e_{m+1}} \Phi_{e_{m+1}}^{-1}(\frac{\rho}{2} + \frac{1}{2}) + \mu_{e_{m+1,k}} \le \varepsilon \\
& \quad |d_k - [d_{k-1}, \dots, d_{k-N}] A_m^\top| \le \varepsilon \qquad \forall k \in [k_s, \tau_m - 1) \\
& \quad |d_k - [d_{k-1}, \dots, d_{k-N}] A_{m+1}^\top| \le \varepsilon \quad \forall k \in [\tau_m, k_e]
\end{aligned} \tag{5.11}$$

5.4.3 Data Structure Extraction Algorithm

The formulations described in the previous sections are then incorporated into a top-down, sliding window algorithm that builds the robust data structure. This algorithm is described in table 5.1. While the algorithm in table 5.1 is a multitier algorithm, the description here is focused on one tier and can be extrapolated beyond by the reader.

First, the algorithm begins by choosing the beginning point, k_s, and end point, k_e, of the segmentation. The data is partitioned into the initial set of segments about a candidate transition point t and the optimization problem $\mathcal{O}1$ and $\mathcal{O}2$ are solved. Feasible solutions containing models A_1, A_2, and a transition point, τ_1 are saved to a solution structure.

Next one data point is transferred from the longer data segment to the shorter data segment. The optimization is performed again and the resulting solutions are recorded. Iterating this procedure through all possible transition points identifies the locations of dynamics discontinuities, because each time the optimization problems $\mathcal{O}1$ and $\mathcal{O}2$ have a solution, a change in the dynamics is observed. The point chosen as a transition point is the candidate transition point that has the smallest probability of a missed transition. In other words, this is the point at which the dynamic discontinuity is greatest.

Having chosen this transition point, the resulting segments are recorded in the first data structure level, $L = 1$. The procedure is then iterated at the next level, for each of the segments that were identified above. Enforcing the constraint of increasing model fidelity, ε is limited to be smaller than the modeling error bound of the parent segment from the previous level.

This partitioning procedure continues for successive levels until no more transition points can be found. When this happens, the next tier of the structure is started. This loop continues until all transition points on all data tiers are identified. The resulting data structure has L levels and N_{ML} segments per level.

5.4.4 In-Sequence Localization Procedure

The in-sequence localization procedure used in this chapter is an evolution from the previously used matching procedure. While in previous work the localization procedure was performed for a fixed size data sequence, the matching procedure illustrated here automatically determines the length of data sequence needed for algorithm convergence. This approach illustrates the strengths, weaknesses, and limitations of the proposed data representation and structuring.

In short, following the initial acquisition of a small set of data, the initial data matrices $\mathbf{z}_{tp}$ and Ω are formed. Using these matrices, model agreement for each model in the data structure is evaluated using equation (5.2). Models that are feasible are retained in a tree structure and models that are infeasible are discarded. With each consecutive acquired data point, the models are re-evaluated and the result tree is redrawn. The result tree itself is flexible, maintaining feasible paths instead of simply feasible models. Because each feasible model represents a range of possible data locations, this means that several branches could contain the same model but different data sequence start points. The localization procedure ends when a single feasible segment remains at the last level of the data structure. The resulting error of localization is thus the uncertainty in data sequence start points among all surviving paths in the resulting tree. This localization procedure is illustrated in more detail in table 5.2.

5.4.5 Online Noise Mitigation During Localization

The dynamical models that are extracted for this work are optimized at transition points to minimize the probability of error. However, in order to maintain a low complexity of implementation, throughout each segment the model agreement is evaluated in a deterministic fashion[2]. This can lead to model agreement errors on the segment following the addition of Gaussian noise. To mitigate this effect during localization, each acquired data point is expected to have a bounded perturbation $|\eta_k| \leq \eta_B$, where η_B is two times the standard deviation of the expected noise. A model is then said to agree with the data if a set of constants $\bar{\eta} = [\eta_{k-1}, ...\eta_{k-N-1}]$ can be found such that,

$$\begin{aligned} |d_k - c_{m,1}(d_{k-1} + \eta_{k-1}) + \dots \\ + c_{m,N}(d_{k-N} + \eta_{k-N-1})| \leq \varepsilon_m. \end{aligned} \tag{5.12}$$

Because the probability that a noise sample will exceed the perturbation bound η_B is not zero, there is a small set of noise realization for which the localization

[2]More elaborate (and more computationally complex) approaches that include information about the noise can also be developed. However, we leave this promising area of research to future work.

procedure will eliminate the correct model. When this happens, it rapidly leads to the elimination of all models on the particular level where the correct model was removed. In this event the tree structure is designed to re-activate all segments on the eliminated level and repeat localization procedure.

5.5 Simulation Results Demonstrating the Robust Model-Based Data Structure Effectiveness

The approach presented in this chapter is tested in three applications. First, continuing the development of a vehicle localization approach using dynamical models, section 5.5.1 presents results using inertial vehicle data. Then section 5.5.2 expands the applications for this approach to economic forecasting using real-life inflation data. Finally, section 5.5.3 demonstrates that our approach can be applied to any set of data by applying the algorithms to data generated using a random walk.

5.5.1 Vehicle Data

Using a data set of inertial vehicle data, the data sequence to be localized is corrupted with 12dB of noise corresponding to the fidelity of a mid-priced sensor [146]. Then a series of experiments starting at a random point are conducted. Each experiment performs the in-sequence localization procedure until a match has been determined. A match is defined as a single feasible segment at the lowest level of the tree structure and a matching error that is less than a pre-determined error bound - for the vehicle localization experiments, that bound was 100 m. In other words, the localization length depends on the initial start point, and not a predetermined test length chosen by the designer.

The results of these experiments are presented in Fig. 5.4. This figure is presented in a standard format that we will use across different data sets. The top subplots quantify the wall clock convergence time, the localization length expressed in number of localization steps, and the distance to the nearest transition at the localization instant.

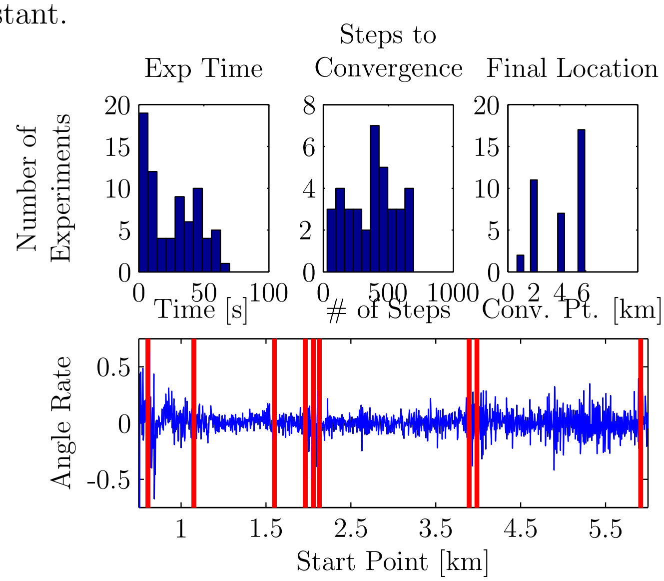

Figure 5.4. In-sequence Localization Results using Vehicle Data using Robust One Dimensional Models.

The subplot representing the computer wall clock time [s] conveys the speed of the in-sequence localization algorithm when using a single 2.66 GHz node of the Penn State super-computing cluster with 4 GB of ram. The subplot showing the localization length demonstrates the number of data sequence points needed to determine the location. The third histogram informs the reader about the final instant of localization when a match has been determined. This plot shows how close the final matching data point is to the nearest transition point, emphasizing the importance of transition points for localization using our model-based approach. To better understand all three plots together, the bottom plot shows the time series data with an overlay of the transition points shown as vertical bars at their respective locations. Note here that the distance between transition points and localization points is typically small, which means that the localization length is directly related to the distance between consecutive transition points on the map. This also means that there will be some road segments that are so plain that localization using this method is not possible.

In the context of vehicle data, the figure demonstrates the speed of convergence of the vehicle location estimate. For example, to determine the length of travel required for localization, the localization length should be multiplied by the map decimation of 5 m. The wall clock convergence time can be interpreted as the maximum speed a vehicle can drive in order for the localization algorithm to keep up with the incoming data. For example, if a vehicle is traveling the 6 km stretch of road described by this map at highway speeds, the vehicle will cross the map in 3 minutes and 45 seconds. Therefore convergence times under 1 minute are more than enough to accommodate vehicle localization. It should be noted that these times are dependent on the size of the map on which localization is occurring, opening an interesting new research avenue of how to best structure millions of data points without slowing down the localization process.

5.5.2 Economic Data

The algorithms developed in the context of vehicle localization need not be constrained only to this application. One interesting application area is economic data that is used by economists and traders to inform decisions on policy and business. Fig. 5.5, using the same format as before, shows the localization results using the monthly consumer price index (CPI) time series[3]. In these experiments, the data is corrupted with 12dB for consistency among our experiments. Heading the importance of transition points, all trials begin before point 400 m in the data to ensure at least one transition point in the data sequence. There are two important

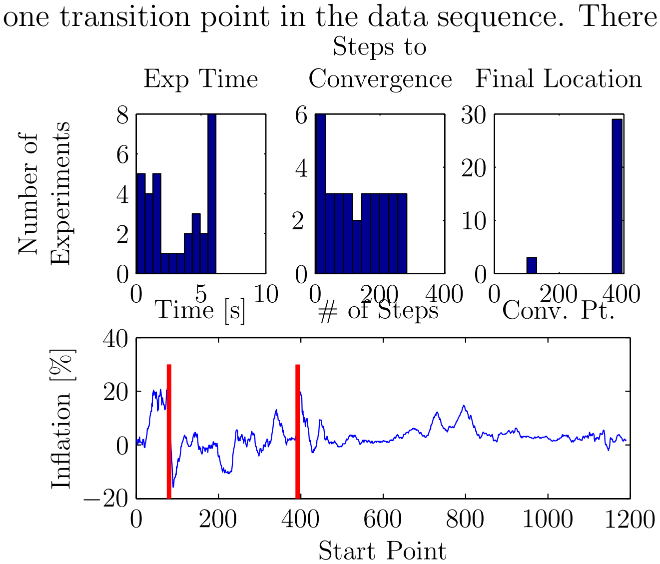

Figure 5.5. In-sequence Localization Results using Economic Data and Robust One Dimensional Data Models.

[3]This data is publicly available, ex. see http://www.usinflationcalculator.com/ for a complete table of monthly inflation numbers from 1914-present.

observations in this figure. First, using the same constants as the vehicle data, the probability of a missed transition point was constrained to 2%. The resulting data structure is quite small, with only 4 significant transition point locations. This is a consequence of the dynamics in the data that invites the question, in physical data is there a significance to the transition points that are identified using our segmentation algorithm? This idea will be further discussed in section 5.6.3. Second, in data sets with few transition points, passing through even a single transition point is sufficient for localization. Note here that all trials converged at the passing of a transition point.

5.5.3 Random Data

Lastly, in Fig. 5.6 which is shown in the same format, we show that this approach can work even if the underlying phenomena is truly random. Here the data set is a Gaussian random walk generated in MATLAB with a standard deviation of 1. Applying the 12dB, the resulting convergence plots are shown below. Note that fixing the probability of a missed transition to 2% yielded a sufficiently fine segmentation for in-sequence localization over the entire data set. One interesting observation about this data set is that while many experiments converge closely to a transition point, a sizable number of experiments do not. We did not observe this in vehicle data or CPI data. This suggests that there may be underlying physical phenomena that change at transition points in the real-world data that do not occur in random data. A possible avenue of future research is to determine which data attributes influence convergence outside of transition points, as this may be a source of discovery of the significance of underlying dynamics in physical data sets.

5.6 Discussion

The preceding numerical results highlight the strengths of the presented approach to data structuring and representation. It is notable that despite the low number of transition points in some sets such as the economic data, a data sequence starting at any point preceding the last transition point was correctly identified.

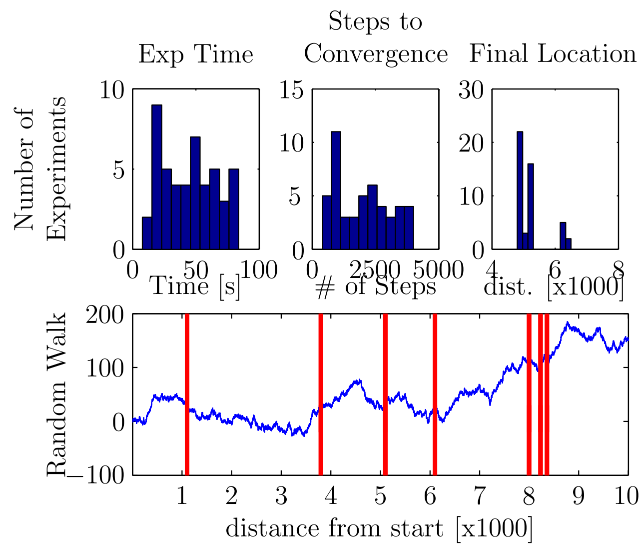

Figure 5.6. In-sequence Localization Results using Random Data using Robust One Dimensional Data Models.

5.6.1 Compactness of Data Representation

The strength of the approach in compacting the data is evident when comparing the new data representation to past results. In particular for vehicle data, when comparing the size of the data representation in this chapter with the size of the data representation in Chapter 4 [23], we observed a 16 fold reduction in the number of extracted models. In addition each model is designed to be more robust to noise so the result is a lower computational burden and a faster convergence time. These strengths are shown in Fig. 5.7 using vehicle data.

This figure shows the wall clock time of localizing a fixed length, noisy data sequence with a random start point to a pre-established dynamical model data structure. The histograms in the figure depict the number of experiments that were completed at the corresponding time, which is measured in seconds. The grey bars demonstrate the performance of a data structure created using the probabilistic optimization of transition points, while the blue bars demonstrate the performance of the data structure constructed for [23]. Using the data structure and representation that is probabilistically optimized for stochastic noise, we observe a two to three order of magnitude decrease in the convergence time versus a data structure that is iteratively built for localization. While the actual improvement may vary from one set to another, this test case demonstrates the inherent advantages of our quality measure on the extracted models.

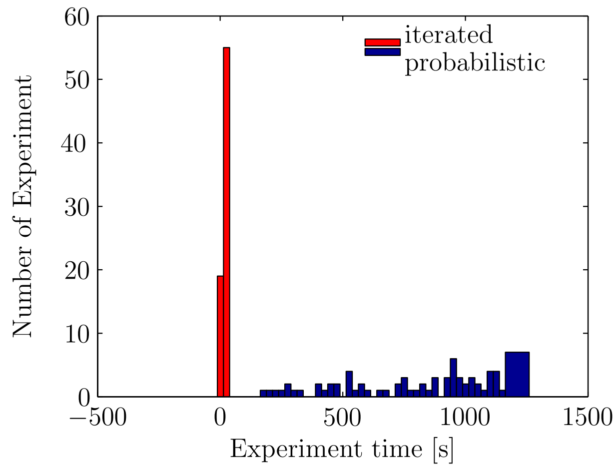

Figure 5.7. Experiment Run Time Comparison for Localization Maps Created in Chapters 4 and 5.

5.6.2 Uniqueness of Transition Points

The strength of the approach in correctly localizing variable length data sequences rests on the uniqueness of the extracted transition points, i.e. the significance of the change in the underlying data dynamics that is observed at each such transition. For the vehicle data, Fig. 5.8 illustrates this significance by plotting the uniqueness of each transition point on the map at the respective transition point location. The figure shows the transition points separated by tier to emphasize that higher tiers are inherently more robust because of the reduced standard deviation of the Gaussian noise.

To create this figure, transition point uniqueness is determined by counting the number of transitions that are satisfied by a given data sequence. For example, taking the data sequence that satisfies the first transition point on the second level, each transition point in the first tier is tested. If a subsequent transition point criteria is satisfied by this data sequence, then the number of similar transition points, tr_s, is increased by one.

Normalizing this metric, the number of similar transitions is divided by the total number of transitions, tr_T, on the tier. Furthermore, to express the number of unique transitions, the ratio tr_s/tr_T is subtracted from one, $1 - tr_s/tr_T$. Then from Fig. 5.8, one can see that both of the transitions on tier 1 are unique. Then on tier 2, the high uniqueness scores show that at most a few transitions have one to two other similar points.

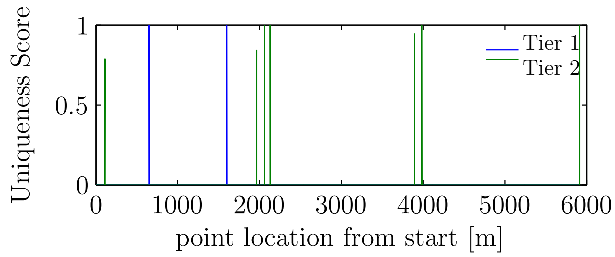

Figure 5.8. Uniqueness of Robust Transition Points in Vehicle Data

5.6.3 Physical Interpretation of Transition Points

Given the small number of unique transition points, it is interesting to ask whether each point carries with it a physical significance in real-world data. Although this was not the object of this study, it appears that at least in the CPI data the transition points are correlated to significant financial events in US history.

For example, the US government began collecting data on the consumer price index in 1914. Our first transition point was detected 80 months later, which roughly corresponds to the sharp financial depression in the US following WWI. This depression was followed by strong growth, and a second depression in 1923, corresponding to transition point 2, occurring about 110 months after January of 1914. The same pattern is observed following the end of WWII: a deep depression resulting from spending restraint following the war results in a transition point in our data at 392 months. The US economy then began growing in 1947, which corresponds to the final transition point.

Similar comparisons can be made in the vehicle data collected by some of the authors over 6000 miles of US roadways. In particular vehicle pitch was observed to be correlated with road construction times, construction equipment, and the actual construction company. This type of information could prove to be useful when building larger maps that require more information for localization.

5.6.4 Limitations

As with any approach and algorithm there are some limitations that affect the performance of the presented algorithm. In the case of this dynamical model-based localization approach, the performance depends strongly on the interplay between the data dynamics, the chosen model order, and the user defined probability of success at a transition point. In other words, the dynamics of the data must exhibit periodic changes that can be quantified using various model orders. For example, in the vehicle data it was determined that a model order of five produced transition points throughout the time series, allowing data sequences to start almost anywhere on the map. Similar results were found in the random data. However, the economic data showed that only four transitions could be found that satisfy the probability criteria. This implies that the underlying dynamics of the economic data set are of low order and not changing sufficiently to produce transitions of high probability throughout the data set.

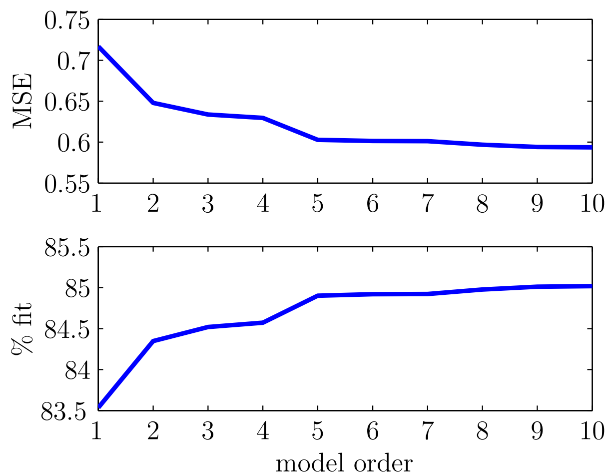

Figure 5.9. Inflation Data Model Order Fit for Robust One Dimensional ARX Models.

Fig. 5.9 shows a closer examination of the inflation data. To generate this figure, a model of each order was generated using the *ar* command in MATLAB, which chooses the model to minimize the mean square error (MSE) or a similar criterion. The figure represents the collected percent fit and MSE from the command for ten different model orders. We use percent fit because it is the standard MATLAB method of comparing data sets. Given that $\mu_{x_{tp}}$ is the data mean of a given segment, percent fit is defined as:

$$\%fit = 1 - \frac{\|\tilde{\mathbf{z}}_{tp} - \mathbf{z}_{tp}\|_2}{\|\tilde{\mathbf{z}}_{tp} - \mu_{x_{tp}}\|_2}. \tag{5.13}$$

Observe that the percent fit and MSE are relatively unchanged across model orders. This suggests that the underlying data dynamics are of very low order and do not vary sufficiently throughout the data set. Thus, for in-sequence localization on the whole data set, the approach suggested in this thesis is not an optimal approach to building a data representation. However, in light of the physical meaning of the transition point, our approach could be used to determine probabilistically important events in time.

5.7 Conclusions and Future Development

This chapter addressed the requirement that data representations used for in-sequence localization need to be robust to the presence of noise in the data collection process. To accommodate the manifestations of noise, a probabilistic approach to model extraction was taken, where models were determined by formulating a stochastic programming approach. In this approach the probability of success about each transition point was lower bounded and appropriate models were determined. In addition, this chapter demonstrated the use of the internal model principle to mitigate the effects of a known sinusoidal noise.

When compared to the model representations from the previous chapter, the identified models here were more robust, and the corresponding transition points were more readily tied to real-world events. The final chapter of this thesis will build on this work and create a multi-dimensional representation whose identification is again formulated in the chance constrained framework. Developing multi-dimensional representations is critical to ensure robustness, accuracy and redundancy for in-sequence localization applications.

Reference Map Creation Algorithm
Initialization
$L = 0$
$\tau_0 = 1$
$\tau_1 = NML$
$tierflag = 0$
σ_η^2 = to expected noise level
FOR tier = 1:Max tier
obtain data at the tier resolution
WHILE $tierflag == 0$
$L = L + 1$
FOR m = 1:N_{ML}
$n = 0$
$k_0 = \tau_{m-1}$ + N
$k_e = \tau_m$
$\tau_n = k_0$
FOR t = k_0:k_e
Solve $\mathcal{O}1$
IF $\mathcal{O}1$ is feasible
record t, A_m, A_{m+1} in a structure S
iterate the transition point counter: n = n + 1
END IF
Solve $\mathcal{O}2$
IF $\mathcal{O}2$ is feasible
record t, A_m, A_{m+1} in a structure S
iterate the transition point counter: n = n + 1
END IF
END FOR
IF n == 0 && $m == M$
$tierflag = 1$
ELSEIF $m < M$ && n > 0
add n to the total number of transitions on the level
from S choose τ_n s.t. τ_n minimizes $\mathbb{P}_{error}$
record the new segments on L
END IF
END WHILE
END FOR

Table 5.1. Data Structure Extraction Algorithm for Robust One Dimensional Models

In-Sequence Localization Algorithm
Initialization
Collect $N+1$ data points:
set loop index: k $= N+1$
initialize data matrices: x_{T_j} and X_j
Begin Localization Loop
FOR l = 1:L
FOR m = 1:$M_L - 1$
IF the parent segment ofA_m is feasible:
Find a set $\bar{\eta}_m$ s.t. $\lvert e_{m,k} \rvert \leq \varepsilon_m$
Find a set $\bar{\eta}_{m+1}$ s.t. $\lvert e_{m+1,k} \rvert \leq \varepsilon_{m+1}$
IF $\bar{\eta}_m$ exists
A_m is feasible, add to its counter
add the m^{th} segment to the result tree
ELSEIF $\bar{\eta}_m$ does not exist AND $\bar{\eta}_{m+1}$ exists
then the data point d_k is a transition point
add the m^{th} segment to the result tree in case of error,
add the $m+1^{st}$ segment to the result tree,
ELSE
remove the m^{th} segment from the result tree
END IF
END IF
END FOR
END FOR
Collect an additional data point
iterate the index: k = k + 1
iterate the data matrices: x_{T_j} and X_j
Following Each Detected Transition Point
FOR l = 1:L
FOR m = 1:N_{ML}
IF (τ_m belongs in the segment &&
segment m is not in the result tree)
add the m^{th} segment to the result tree
END IF
END FOR
END FOR

Table 5.2. In-Sequence Localization Algorithm for Model Structures Using Robust One Dimensional Models.

Chapter 6

Robust Data Structures for In-Sequence Localization using Multiple Time Series

This chapter builds on the in-sequence localization data representations developed in chapters 4 and 5. In these previous chapters, ARX models were introduced as data representations for the application of in-sequence localization. The models were then tuned to improve the robustness of the localization process in the presence of noise that was both deterministic and stochastic. By extracting robust model representations, the mitigation of noise is off-loaded onto the map-building process (which is usually performed in an environment that has unlimited computational power).

In this chapter, the linear model representations are extended to include multiple time series. The particular case is considered where the observed time series are correlated and act as "attributes" of an unknown process to be inferred after localization. For example, this situation arises in vehicle localization, when multiple data time series are collected from several sensors that describe attributes of the road. The inclusion of multiple parallel time series is shown to increase the speed of in-sequence localization. Furthermore, through several examples, the domain of the approach is expanded to include environmental health monitoring and energy generation management. Including these examples demonstrates the versatility of the approach presented in this thesis and opens new avenues of post-dissertation research.

Preliminary work in this chapter was presented in the 2014 Penn State College of Engineering Research Symposium [26], where the work was awarded the best paper award. In addition, further work will be presented in the 2014 IEEE Conference on Decision and Control. A journal manuscript of the complete chapter is currently under development for submission to IEEE Transactions on Knowledge and Data Engineering.

6.1 Introduction

Tailoring data representations to the problem in-sequence localization, the previous chapters in this thesis proposed the use of linear dynamical models to reduce the dimension of stored data and enable rapid in-sequence localization (see Chapter 4 or publications [20, 22]). These models can be further tuned to increase the robustness of the representations to certain quantified and known process noises (see Chapter 5 or publications [24, 23]). However, the approach to data representation and in-sequence localization in these chapters was limited to one-dimensional time series such that a reference data sequence was mapped and new noisy data from this sequence was used for localization within the reference sequence. Continuing the development of this work, this chapter presents a self-contained description of the design of a multiple time series data representation that is used for in-sequence localization. This representation retains the properties of dimension reduction, computational simplicity in localization and noise robustness that are necessary for in-sequence localization on a mobile platform.

Because we find the application of vehicle localization to be most intuitive, the work presented here often refers to this application to convey concepts and ideas. However, the solutions proposed here are not limited to this domain. In fact, results are presented in two other applications: variable generation forecasting [193, 194], and stream health monitoring [195, 196, 197]. Thus the numerical experiments hint at future work that will be performed in subsequent publications.

6.1.1 Vehicle Localization Using Multiple Time Series

The previous chapters in this thesis proposed a dynamical model-based solution to the problem of in-sequence localization, where dynamical models are used to represent the data and the model predictions are used to eliminate feasible location regions. This work demonstrated the ability of IMU data to localize a vehicle and also demonstrated the limitation of using a single dimension of the IMU sensor for localization (see Chapter 5), namely the relatively small subset of points on any given map that are robust to noise. Thus, to implement the approach on a practical vehicle, the localization maps must be generated using multiple sensors that complement each other, adding information to the localization process in regions of the map that are at best complimentary and, at worst, only overlapping with a few other sensors. This chapter presents one approach to building multiple-time series representations for the purpose of building localization maps for vehicle localization.

In the realm of vehicle localization, multiple time series can have in one of two configurations with respect to real world road geometry: the time series could represent the data from a single sensor on multiple roads, or the time series could represent the sensory output from multiple sensors simultaneously measuring the same road. To make the description clearer, the former case is termed multi-dimensional data because one can imagine the 3-dimensional world in which roads exist, and the latter case is termed multi-attribute data because the output of each sensor is an attribute of the particular road that is being traversed. These attributes are related to one another via the spatial location of the readings in the world.

The focus of this chapter is multiple-attribute time series, leaving the development of multi-dimensional time series structures to future work. The multiple-attribute time series representation presented here can accommodate any number of time series necessary to ensure highly accurate maps. The following is a more precise definition of in-sequence localization using multiple-attribute time series. In this definition, the number of time series is denoted by N_{ts} and the particular time series is indexed by γ. As defined in Chapter 3, the length of the complete historical time series is i, and the length of the noisy subset collected some time T later is $k+1$. The problem of multi-attribute time series in-sequence localization can now be mathematically defined.

Multi-attribute time series in-sequence localization: Given a multi-attribute time series with N_{ts}-component time series ($\mathbf{D}_1$,$\mathbf{D}_2$,...,$\mathbf{D}_{N_{ts}}$) $\forall\gamma = 1 : N_{ts}$, where $\mathbf{D}_\gamma = \{d_{\gamma,1}, d_{\gamma,2}, \ldots, d_{\gamma,i}\}$, and a noisy subset (of length $k+1$) of the series collected with some time delay T, $\bar{\mathbf{x}} = \{d_{\gamma,T} + \eta_{\gamma,T}, d_{\gamma,T+1} + \eta_{T+1}, \ldots, d_{\gamma,T+k} + \eta_{\gamma,T+k}\}$ $\forall\gamma = 1 : N_{ts}$, find the index j such that the subset of the original time series $\bar{\mathbf{d}} = \{d_{\gamma,j}, d_{\gamma,j+1}, \ldots, d_{\gamma,j+k}\}$ $\forall\gamma = 1 : N_{ts}$ most closely matches the collected time series subset.

6.2 Multi-Dimensional Data Representations Survey

This section overviews some work published in the domain of data representation for subsequence matching. This area is the closest published analogue to in-sequence localization.

6.2.1 Multi-Attribute Time Series

In the literature, multi-attribute time series research has focused heavily on clustering. For example Povinelli and Feng [81] develop a clustering algorithm for stock volume and price data. This algorithm concatenates the data attributes and forms clusters in the data phase space. Another clustering approach by Kahveci et. al. [82] focuses on the identification of shift and scale invariant clusters in multi-attribute time series. Uniqueness in the clustered patterns is addressed by Lee et. al. [83]. This uniqueness is obtained by stacking 1-dimensional patterns to find the smallest unique multidimensional representation.

6.2.2 Multidimensional Time Series

Some authors have focused on discovering patterns in truly multidimensional time series. An interesting subproblem here is determining the number of useful dimensions. Minnen et. al. [84] address this problem of identifying sub-dimensional patterns with an efficient algorithm. Simultaneous work by these authors [85] also explores the use of Hidden Markov Models (HMM) in the multidimensional motif discovery. An interesting aspect of this work is that it is scalable from one to many dimensions.

6.2.3 Clustering

Both works by Minnen et. al. [84, 85] follow the general pattern of published work addressing clustering. In addition, published results by Plant et. al. [86] focus specifically on clustering multidimensional time series based on the interaction between the dimensions. This is an interesting approach because it preserves additional information in multiple dimensions.

Clustering can also be based on temporal similarities between frequency patterns, as shown by Tatavarty et. al. [87]. In the spectral domain, i.e. using the eigenvalues of the similarity matrix, clustering is demonstrated by Wang et. al. [88]. Key aspects of clustering such as similarity searching and structuring are addressed by Matsubara et. al. [90] and Moreira et. al. [91] respectively.

Yet another interesting application of multidimensional time series analysis is prediction. Shibuya et. al. [89] and the references within demonstrate the use of multidimensional time series to predict behavior. A key component in this research is determining the causality of one dimension on another. In other words, researchers are looking to determine whether the fluctuation in one time series can be used to predict the behavior of a second time series. This is yet another method of determining the importance of additional dimensions.

A similar application in vehicular technology is accident prediction by Gonzalez et. al. [92]. In this work, multidimensional traffic data is mined for early anomaly detection. These anomalies are used by traffic engineers to detect accidents early and re-route traffic to maintain flow along the nation's highways.

6.3 Dynamical Model Representations of Multi-Attribute Time Series

The multiple-attribute time series are represented using multi-input, multi-output (MIMO) AutoRegressive models with an eXogenous input (MIMO ARX). These models are particularly convenient for the problem of in-sequence localization because they can reduce the size of the stored data, they can begin localization at any time inside the domain of the model, and they retain the sequential information that is encoded in the data. In addition, localization procedures based on MIMO ARX models only evaluate the most recently collected data point for localization and thus reduce the computational costs of localization for mobile platforms. Lastly, the model parameters can be tuned to improve the robustness of the representations and the localization mechanism to noise. This tuning will be described in detail in section 6.4. The N^{th} order MIMO ARX model with N_{ts} inputs is described in equations (6.1).

$$\mathbf{z}_n = \mathbf{A}\omega_{n-1} + \mathbf{e}(n) \tag{6.1}$$

$$\mathbf{z}_n = \begin{pmatrix} x_{1,n} \\ \vdots \\ x_{N_{ts},n} \end{pmatrix}$$

$$\mathbf{A} = \begin{pmatrix} c_{1,1_1} & \dots & c_{1,N_1} & \dots & c_{1,1_{N_{ts}}} & \dots & c_{1,N_{N_{ts}}} \\ & \vdots & & \ddots & c_{N_{ts},1_{N_{ts}}} & \dots & c_{N_{ts},N_{N_{ts}}} \end{pmatrix}$$

$$\omega_{n-1} = \begin{pmatrix} x_{1,n-1} & \dots & x_{1,n-N} & \dots & x_{N_{ts},n-1} & \dots & x_{N_{ts},n-N} \end{pmatrix}$$

$$\mathbf{e}(n) = \begin{pmatrix} e_{x_{1,n}} \\ \vdots \\ e_{x_{N_{ts},n}} \end{pmatrix}, \; \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_{x_1} \\ \vdots \\ \varepsilon_{x_{N_{ts}}} \end{pmatrix},$$

where $\mathbf{z}_n$ is the vector of simultaneous measurements described by the model $\mathbf{A}$ using the set of previous measurements ω_{n-1} up to a modeling error, the set of which is collected in the vector $\mathbf{e}(n)$. The modeling errors are bounded to the positive and negative side by modeling error bounds which are collected in the vector $\boldsymbol{\varepsilon}$.

In some of the equations below we refer to the magnitude of the modeling error in each model output. In these cases, the notation $|\mathbf{e}(n)|$ is used to denote a vector of modeling error magnitudes at the n^{th} instant in time such that

$$|\mathbf{e}(n)| \quad = \quad \left(\left|e_{x_{1,n}}\right|, \ldots, \left|e_{x_{N_{ts},n}}\right|\right)^T. \tag{6.2}$$

Describing the equations in short, the n^{th} point in the multi-attribute time series, $\left(x_{1,n}, \ldots, x_{N_{ts},n}\right)^T$, is approximated using the previous N data points. The n^{th} point is referred to as the test point because when the model prediction is evaluated against the modeling error bound, equation (6.2), models that agree with their bound are feasible, and models that do not agree with their error bound are infeasible. The modeling error bound is automatically chosen such that the model describes the data with the tightest possible error bound given the fixed model order. The modeling error bound applies to both the negative and the positive side of the modeling error.

Because the model order is fixed such that at least some data compression is observed and because the modeling error bound should be tight to enable in-sequence localization, multiple sequential models are needed to describe an entire data set. That is, the data set is subdivided into sequential sets which are defined by model agreement or disagreement. In the section to follow, the definition of model agreement in this chapter is clarified and extensions for future work are shown.

6.3.1 Model Agreement

Recall that the points that separate the intervals supporting each model are called transition points and are denoted by τ_m, where m is the transition point index denoting that this transition point belongs to the m^{th} model. Transition points are the points in the data where the underlying dynamics are changing such that the model that previously described the data is no longer valid. In the case of multiple attribute time series this change in dynamics can occur on all time series or a subset of the time series. This change in dynamics (across multiple time series) is described by a model disagreement of the data for the data subset that is ending, and a model agreement for the new data subset that is beginning. Describing the event where all model outputs disagree with the data at a transition point, τ_m, there are $3N_{ts}$ inequalities that hold:

$$\begin{aligned}
\mathbf{e}_m(\tau_m) &= \mathbf{z}_{\tau_m - 1} - \mathbf{A}_m\omega_{n-2}, & \text{s.t.} \quad & |\mathbf{e}_m(\tau_{m-1})| \leq \boldsymbol{\varepsilon} \\
\mathbf{e}_m(\tau_m) &= \mathbf{z}_{\tau_m} - \mathbf{A}_m\omega_{n-1}, & \text{s.t.} \quad & |\mathbf{e}_m(\tau_m)| > \boldsymbol{\varepsilon} \\
\mathbf{e}_{m+1}(\tau_m) &= \mathbf{z}_{\tau_{m+1}} - \mathbf{A}_{m+1}\omega_{n-1}, & \text{s.t.} \quad & |\mathbf{e}_{m+1}(\tau_m)| \leq \boldsymbol{\varepsilon}
\end{aligned} \tag{6.3}$$

In these inequalities $|\mathbf{e}_m(\tau_m)|$ is the vector of the m^{th} model's modeling error magnitudes at the transition point τ_m. The first inequality shows that the m^{th} model agrees with the data in the previous time instant (and all instances in the corresponding data segment). The second inequality shows that at the current data point the m^{th} model does not describe the acquired data. The third inequality shows that the next model, $m + 1^{st}$, agrees with the new data that is being acquired. Together these inequalities demonstrate that the underlying dynamics are changing and that the pair of models that describe the data on each side of the transition point captures this change.

On the other hand in the case where only some of the time series disagree with the models, the transition point inequalities will have the following form below, where the symbol $\gtrless$[1] denotes the fact that some of the time series exceed their modeling error bound and some do not.

$$\begin{aligned}
\mathbf{e}_m(\tau_m) &= \mathbf{z}_{\tau_m - 1} - \mathbf{A}_m\omega_{n-2}, & \text{s.t.} \quad & |\mathbf{e}_m(\tau_{m-1})| \leq \boldsymbol{\varepsilon} \\
\mathbf{e}_m(\tau_m) &= \mathbf{z}_{\tau_m} - \mathbf{A}_m\omega_{n-1}, & \text{s.t.} \quad & |\mathbf{e}_m(\tau_m)| \gtrless \boldsymbol{\varepsilon} \\
\mathbf{e}_{m+1}(\tau_m) &= \mathbf{z}_{\tau_{m+1}} - \mathbf{A}_{m+1}\omega_{n-1}, & \text{s.t.} \quad & |\mathbf{e}_{m+1}(\tau_m)| \leq \boldsymbol{\varepsilon}
\end{aligned} \tag{6.4}$$

This latter form of the equations is more complex to implement but more applicable in cases where a large number of uncorrelated time series are used. This is because as the number of time series increases, one can expect that the number of transition points as defined by equation (6.3) will decrease.

Assumption: In this chapter model disagreement is defined as shown in equation (6.3). In essence, all time series are required to disagree with a model simultaneously at each transition point τ_m.

[1] As defined in the symbols section, the symbol $\gtrless$ when used with vector quantities implies that some of the row-wise inequalities are $>$, some are $<$.

This is a simplifying assumption that reduces the computational effort in the creation of the in-sequence localization maps by simplifying the solution space and allowing faster computation than the more complex (6.4). The goal of reducing this computational effort is to increase the amount of incoming data that can be processed and stored. However, it is important to note that requiring all time series to disagree rests on the idea that this chapter is developed in the context of multi-attribute time series, i.e. series that are correlated with one another via an underlying process; a correlation readily observed in real data and demonstrated in feasibility in the numerical results of this chapter.

The correlation in the time series leads to a coupling in the models such that even when a transition point is not readily apparent by inspection, a jump in one time series leads to a shift in the model parameters of the rest of the time series. Thus the change in dynamics in one time series will trigger a shift in the determined data model such that a transition point is generated across all time series. This should be true provided that the time series are sufficiently coupled, and judiciously chosen. However, as the number of time series grows and the time series become more independent, we expect that it may be increasingly difficult to find transition points. This will be further discussed at the end of the chapter in the discussion section.

Thus the assumption above simplifies the problem to be solved at the cost of limiting the number and independence of the time series. However, we feel that this should not be a problem because this work is devoted to multiple attribute time series which should be correlated. As seen later in the chapter, for a well chosen set of localization sensors, an abundance of transition points is found and the resulting in-sequence localization maps perform well given our assumptions.

6.4 Defining Robustness at Transition Points

Transition points are critically important for in-sequence localization because they provide precise locations within the data. In contrast, a feasible model only shows a possible set of locations. For this reason, this chapter and the previous chapter focus on in-sequence localization using dynamical models whose underlying data is determined by finding the most robust transitions. In essence, these are the transition points at which model transitions are not obscured by the presence of noise. To make the discussion below more general, assume that the m^{th} model is that whose data segment is ending and the $m+1^{st}$ model is the model whose data segment is beginning.

The transition point at which the m^{th} model segment is ending is the point at which new dynamics are seen in the observed data, and these dynamics are being represented by the $m+1^{st}$ model. Practically this means that the modeling error of the m^{th} model exceeds its error bound $\boldsymbol{\varepsilon}$, while the modeling error of the $m+1^{st}$ model agrees with its error bound $\boldsymbol{\varepsilon}$. This problem is defined by the inequalities[2] as follows. Given two data adjoining data intervals $[\tau_{m-1}, \tau_m]$ and $[\tau_m, \tau_{m+1}]$, we would like to find two models that disagree[3] at the transition point τ_m,

$$\begin{aligned} \mathbf{e}_m(\tau_m) &= \mathbf{z}_{\tau_m} - \mathbf{A}_m \omega_{n-1}, && \text{s.t.} \quad |\mathbf{e}_m(\tau_m)| > \boldsymbol{\varepsilon} \\ \mathbf{e}_{m+1}(\tau_m) &= \mathbf{z}_{\tau_m} - \mathbf{A}_{m+1} \omega_{n-1}, && \text{s.t.} \quad |\mathbf{e}_{m+1}(\tau_m)| \leq \boldsymbol{\varepsilon} \end{aligned} \tag{6.5}$$

but agree with the data on their respective domains as represented by the data indices t in the following equations,

$$\begin{aligned} \mathbf{e}_m(t) &= \mathbf{z}_{t-1} - \mathbf{A}_m \omega_{t-2}, && \text{s.t.} \quad |\mathbf{e}_m(t)| \leq \boldsymbol{\varepsilon} \\ &\text{and} \\ \mathbf{e}_{m+1}(t) &= \mathbf{z}_t - \mathbf{A}_{m+1} \omega_{t-1}, && \text{s.t.} \quad |\mathbf{e}_{m+1}(t)| \leq \boldsymbol{\varepsilon} \end{aligned} \tag{6.6}$$

In the presence of noise, the output of each model is stochastic. In particular, in the presence of the Gaussian noise, $\mathcal{N}(0, \sigma_\eta)$, observed in vehicle IMU data, the output error of each model is Gaussian with a mean, μ_m, that corresponds to the modeling error, $\mathbf{e}_m$,

$$\begin{aligned} \mathbf{e}_m(t) &= \mathbf{z}_t - \mathbf{A}_m \omega_{t-1} \\ \mathbf{e}_{m+1}(t) &= \mathbf{z}_t - \mathbf{A}_{m+1} \omega_{t-1}. \end{aligned} \tag{6.7}$$

Then assuming that the noise component of each time series is independent[4] of the

[2]These inequalities have corresponding polynomial equivalents.

[3]One model fits the data, and the other does not.

[4]The constraint of noise independence can be relaxed by modeling correlated noise as the

noise component of other time series, and that the standard deviation of noise, σ_η, has been normalized[5], then the covariance matrix, Σ_m, is related to the model by the squared sum of the coefficients,

$$\begin{aligned}\mathbf{\Sigma}_m &= (I + \mathbf{A}_m \cdot \mathbf{A}_m^\top) \cdot \sigma_\eta^2 \\ \mathbf{\Sigma}_{m+1} &= (I + \mathbf{A}_{m+1} \cdot \mathbf{A}_{m+1}^\top) \cdot \sigma_\eta^2.\end{aligned} \tag{6.8}$$

Thus the events at a transition point $|\mathbf{z}_{\tau_m} - \mathbf{A}_m \omega_{\tau_m - 1}| > \boldsymbol{\varepsilon}$ and $|\mathbf{z}_{\tau_m} - \mathbf{A}_{m+1} \omega_{\tau_m - 1}| \leq \boldsymbol{\varepsilon}$ are described by the probabilities $\mathbb{P}(|\mathbf{e}_m(\tau_m)| > \boldsymbol{\varepsilon})$ and $\mathbb{P}(|\mathbf{e}_{m+1}(\tau_m)| \leq \boldsymbol{\varepsilon})$, respectively, where the distribution function of the error is multi-variate Gaussian with an order equal to the number of time series used.

Consider the case of two time series, the 2-dimensional MIMO ARX models at a transition point, whose errors are described by bivariate Gaussian distributions that can be visualized as shown in Fig. 6.1. Visualizing the distributions in two dimensions illustrates the source of error when detecting a transition point, that is the region below the distributions about the modeling error bound $\boldsymbol{\varepsilon}$ where the distributions overlap. Integrating over this region gives the probability of making an erroneous decision about the transition point. This type of reasoning has long been employed in radar detection theory, where the overlap in one-dimensional Gaussian distributions is used to determine the optimal threshold location [190].

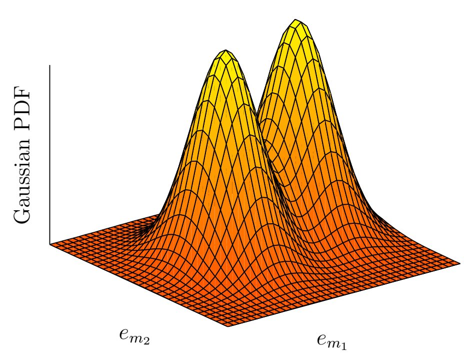

Figure 6.1. Example of the Uncertainty in Transition Points, Represented by the Gaussian Distribution Overlap In Two Dimensions.

model output of a MIMO model whose input is i.i.d. noise.

[5]Each time series has been divided by the standard deviation of the noise. The reason for this is described in the discussion section.

To minimize the probability of an error at a transition point in the presence of noise, the two probability distributions must be maximally separated and the covariance matrix of each distribution must be minimized. For a known noise description, these distributions can be shaped during model extraction by choosing the m^{th} model such that its model output error in all dimensions is maximized at the transition point, choosing the $m+1^{st}$ model such that its model output error is minimized at the transition point, and choosing both models such that their respective covariance matrices, $\boldsymbol{\Sigma}_m$ and $\boldsymbol{\Sigma}_{m+1}$, are minimized. Described in terms of probabilities, a set of models that maximizes the probability of correct detection at a transition is such that the probability of the m^{th} model output error exceeding $\boldsymbol{\varepsilon}$ is large, and the probability of the $m+1^{st}$ model output error being less than $\boldsymbol{\varepsilon}$ is large. These probabilities, however, are not independent because the data vector that is input into both models is identical. For this reason we want to maximize the joint probability of success at a transition point.

6.4.1 Mathematical Formulation of the Model Extraction Problem

Combining the probabilistic description of the transition point events with the inequalities that define model agreement on the intervals about the transition point results in the following problem formulation, that is needed to find the models $\mathbf{A_m}$ and $\mathbf{A_{m+1}}$.

Problem Formulation used in this Chapter:

$$\begin{aligned}
&\underset{\mathbf{A_m},\mathbf{A_{m+1}},\boldsymbol{\varepsilon}}{\text{maximize}} \quad \mathbb{P}(|\mathbf{e}_m(\tau_m)| > \boldsymbol{\varepsilon}, |\mathbf{e}_{m+1}(\tau_m)| \leq \boldsymbol{\varepsilon}) \\
&\text{subject to} \\
&\qquad \mathbf{e}_m(t) = \mathbf{z}_{t-1} - \mathbf{A}_m\omega_{t-2}, \qquad \text{s.t. } |\mathbf{e}_m(t)| \leq \boldsymbol{\varepsilon},\ \forall t \in [\tau_{m-1}, \tau_m) \\
&\qquad \mathbf{e}_{m+1}(t) = \mathbf{z}_t - \mathbf{A}_{m+1}\omega_{t-1}, \quad \text{s.t. } |\mathbf{e}_{m+1}(t)| \leq \boldsymbol{\varepsilon},\ \forall t \in [\tau_m, \tau_{m+1})
\end{aligned} \tag{6.9}$$

Now for completeness, the problem described by the transition point inequalities in (6.4) is also formulated. Here κ model outputs (time series) do not agree with their modeling error bounds and $N_{ts}-\kappa$ model outputs agree with their modeling error bounds. Even more importantly, it is unknown prior to searching for transition points which time series will exhibit a shift in dynamics that leads to model disagreement. Then the sets of model row indices denoting the two different model agreements are defined as follows.

$$\tilde{\Pi} = \{\gamma, \quad s.t. \left|\mathbf{e}_{m_\gamma}(\tau_m)\right| = \left|z_{\tau_{m_\gamma}} - [c_{\gamma,N_1} \dots c_{\gamma,1_{N_{ts}}} \dots c_{\gamma,N_{N_{ts}}}]\omega_{\tau_m}\right| > \boldsymbol{\varepsilon}_\gamma\}$$
$$\hat{\Pi} = \{\gamma, \quad s.t. \left|\mathbf{e}_{m_\gamma}(\tau_m)\right| = \left|z_{\tau_{m_\gamma}} - [c_{\gamma,N_1} \dots c_{\gamma,1_{N_{ts}}} \dots c_{\gamma,N_{N_{ts}}}]\omega_{\tau_m}\right| \leq \boldsymbol{\varepsilon}_\gamma\} \quad (6.10)$$

Where $\tilde{\Pi}$ is the set of indices denoting the model outputs whose output exceeds the modeling error bound and $\hat{\Pi}$ is the set of indices that denoting the model outputs whose output agrees with the modeling error bound. Then using these set definitions and equations (6.4), the original problem posed above can be rewritten as follows.

Broader Problem Formulation:

$$\begin{aligned}
\underset{\mathbf{A_m},\mathbf{A_{m+1}},\boldsymbol{\varepsilon}}{\text{maximize}} \quad & \mathbb{P}(|\mathbf{e}_{m_\gamma}(\tau_m)| \leq \boldsymbol{\varepsilon}_\gamma \quad \forall\gamma \in \tilde{\Pi}, \\
& |\mathbf{e}_{m_\gamma}(\tau_m)| > \boldsymbol{\varepsilon}_\gamma \quad \forall\gamma \in \hat{\Pi}, \\
& |\mathbf{e}_{m+1}|(\tau_m)| \leq \boldsymbol{\varepsilon}) \\
\text{subject to} \quad & \qquad\qquad\qquad\qquad\qquad\qquad (6.11) \\
& \mathbf{e}_m(t) = \mathbf{z}_{t-1} - \mathbf{A}_m\omega_{t-2}, \qquad \text{s.t. } |\mathbf{e}_m(t)| \leq \boldsymbol{\varepsilon},\ \forall t \in [\tau_{m-1}, \tau_m) \\
& \mathbf{e}_{m+1}(t) = \mathbf{z}_t - \mathbf{A}_{m+1}\omega_{t-1}, \quad \text{s.t. } |\mathbf{e}_{m+1}(t)| \leq \boldsymbol{\varepsilon},\ \forall t \in [\tau_m, \tau_{m+1})
\end{aligned}$$

6.5 Identifying Models with Robust Transition Points

Ideally, we would like to pose the problem of identifying robust transition points as a convex problem so that we can take advantage of the reliable, efficient and fast solving method that are available. However, as posed above, neither problem (6.9) nor problem (6.11) are convex. This is because the problems are posed as several coupled inequalities, where the coupling occurs through the noise. Moreover, in each problem the realizations of the stochastic noise multiply the decision variables [198]. A further complication is the fact that in contrast to previously published work [199], the noise distribution is defined via the decision variables during the problem solution, and is therefore initially unavailable.

Despite the high level of complexity in this problem, there exist approaches that are capable of finding exact solutions. One approach developed by Jasour et. al. [200, 201] finds a solution to the problem in terms of moments in the space of measures, rather than explicitly in the space of model coefficients. The approach is briefly described below and the reader is directed to the cited publication for more detail.

The approach developed by Jasour et. al., while optimal in the sense that exact solution can be found, is computationally expensive. For this reason in this thesis, a more computationally efficient approximation is sought such that the models and transition points can be identified across increasingly larger time series. This approximation is described in this chapter beginning with section 6.5.2.

6.5.1 An Asymptotically Optimal Solution Approach

Solving a highly non-convex stochastic programming problem using the approach in [200, 201] expands on the idea that the solution of such a problem can be found if the problem is reformulated as an expected value problem. In essence, the objective function of the problem is reformulated in terms of the probabilistic moments, and solutions are numerically found in the moment space and then converted to solutions of the original problem.

To find the solution, the designer first chooses the number of moments in the approximation. Choosing the number of moments is important because choosing large numbers of moments leads to an exact solution of the problem making this approach asymptotically optimal. However for a large number of moments the approach is computationally expensive, reducing the maximum practically feasible size of the data set.

The moments are arranged in a vector of multidimensional moments that characterizes a probability measure, μ_p. This probability measure has a support set, K, that is defined by the parameters of the problem - in this case the uncertainty, η, the models $\mathbf{A_m}$ and $\mathbf{A_{m+1}}$, the modeling error bounds $\boldsymbol{\varepsilon}$, and the linear constraints above. Then using the vector of moments and the support set K, a localization matrix and a moment matrix are created. Now the original problem is restated as a semidefinite programming (SDP) problem (shown below in equation (6.12)) whose solutions are a positive definite moment matrix, MM, and a positive definite localization matrix, LM. In essence, in the moment space, we maximize the zeroth moment, α_0, of the measure μ_p such that we can find a positive semi-definite localization matrix and a positive semi-definite moment matrix. This approach is proved and described in greater detail in [202, 203].

$$\begin{aligned} &\underset{LM,MM}{\text{maximize}} \quad \alpha_0 \\ &\text{subject to} \\ &\qquad LM \geq 0, MM \geq 0 \end{aligned} \tag{6.12}$$

Once a solution to the problem is found, the decision variables (linear model coefficients in this thesis) can then be extracted from the moment vector. In the case where the moment vector belongs to a set of feasible solutions, an appropriate solution vector is chosen based on the parameters of the problem being solved. The full description of this approach and the attached proofs can be found in [200, 201].

6.5.2 An Approximate Solution Approach

While the approach above leads to an exact solution for large numbers of moments, it is computationally expensive (i.e. the number of computations grows quickly with the size of the data set). This is further complicated by the fact that the problem must be solved for each possible transition point on the whole data set. As the data set scales to a size such as the one needed to describe an entire road network, the cost of this algorithm becomes prohibitive. For this reason, this text proposes a faster approximation, such that the problem can be solved iteratively and the data set size can be increased.

6.5.3 Approximation Procedure

The procedure to obtain the approximation to problem (6.9) is complex and for this reason it is outlined here prior to explaining the details of this chapter. The first step is to approximate the objective function by describing the uncertainty using an approximated uncertainty model and taking the log of the resulting probabilistic expression. Optimizing over the approximation of the objective function requires the identification of several preliminary parameters including the covariance of the approximated noise term, and the model agreement bound for the expected models. To obtain estimates for these quantities, the problem is idealized using the approach outlined in Chapter 5, and the rows of the MIMO models are determined separately using the procedure outlined in section 5.4. Lastly, the solution space of the problem is divided into convex subsets, each of which is searched for a solution of the approximated problem.

The table below summarizes the approximation steps that will be described in the sections to follow. The approximation is also followed by a summary of implementation to guide the reader.

Building the Problem Approximation
1. Approximate the objective function by using an approximated noise model in the right hand side of the objective function inequalities, in problem (6.9), and taking the log of the resulting expression (section 6.5.4).
2. Design a procedure to obtain preliminary values for $\boldsymbol{\varepsilon}$ and the covariance of the approximated noise term (section 6.5.5).
3. Divide the solution space into convex subsets (section 6.5.6).

Table 6.1. Steps to Building the Problem Approximation

6.5.4 Approximating the Objective Function

The first step in approximating problem (6.9) is approximating the objective function. A common approach to approximating complex probabilistic expressions is to assume that the events being modeled are independent[6]. In essence the probability of the events of the m^{th} model exceeding its error bound $\boldsymbol{\varepsilon}$, and the $m+1^{st}$ model agreeing with its modeling error bound $\boldsymbol{\varepsilon}$, is approximated by the multiplication of the respective probabilities. This is an approximation because, at a transition point, the same underlying data segment is input into each model. The problem

[6]This step is not strictly necessary, but it opens the path to future implementations using problem formulation (6.11).

below describes the approximation.

$$\begin{aligned}
&\underset{\mathbf{A_m},\mathbf{A_{m+1}},\boldsymbol{\varepsilon}}{\text{maximize}} \quad \mathbb{P}(|\mathbf{e}_m(\tau_m)| > \boldsymbol{\varepsilon}) \cdot \mathbb{P}(|\mathbf{e}_{m+1}(\tau_m)| \leq \boldsymbol{\varepsilon}) \\
&\text{subject to} \\
&\qquad \mathbf{e}_m(t) = \mathbf{z}_{t-1} - \mathbf{A}_m \omega_{t-2}, \qquad \text{s.t. } |\mathbf{e}_m(t)| \leq \boldsymbol{\varepsilon},\ \forall t \in [\tau_{m-1}, \tau_m) \\
&\qquad \mathbf{e}_{m+1}(t) = \mathbf{z}_{t} - \mathbf{A}_{m+1} \omega_{t-1}, \quad \text{s.t. } |\mathbf{e}_{m+1}(t)| \leq \boldsymbol{\varepsilon},\ \forall t \in [\tau_{m}, \tau_{m+1})
\end{aligned} \tag{6.13}$$

This problem is the first approximation to problem (6.9) and it is still not convex because the realizations of the random noise multiply the decision variables. To address this problem, a common approximation is to separate the random terms in the objective and move these terms to the right hand side of the inequality. Here we describe the estimates of the uncertainty (on the right hand side) as vectors ζ_m and ζ_{m+1}. These vectors have Gaussian distributions centered at 0 with covariance matrices Σ_m and Σ_{m+1}, respectively. To arrive at this approximation, first expand the model error matrix terms in the model error bound comparison inequality[7],

$$\begin{aligned}
\mathbf{e}_m(\tau_m) > \boldsymbol{\varepsilon} = \mathbf{z}_{\tau_m} - \mathbf{A}_m \omega_{n-1} > \boldsymbol{\varepsilon} \\
= \begin{pmatrix} d_{1,n} + \eta_{1,n} \\ \vdots \\ d_{N_{ts},n} + \eta_{N_{ts},n} \end{pmatrix} - \mathbf{A}_m \begin{pmatrix} d_{1,n-1} + \eta_{1,n-1} \\ \vdots \\ d_{N_{ts},n-N} + \eta_{N_{ts},n-N} \end{pmatrix} > \boldsymbol{\varepsilon}.
\end{aligned} \tag{6.14}$$

Then using the expanded equations, move all terms containing noise to the right hand side of the model error bound comparison inequality,

$$\left(\begin{pmatrix} d_{1,n} \\ \vdots \\ d_{N_{ts},n} \end{pmatrix} - \mathbf{A}_m \begin{pmatrix} d_{1,n-1} \\ \vdots \\ d_{N_{ts},n-N} \end{pmatrix}\right) + \left(\begin{pmatrix} \eta_{1,n} \\ \vdots \\ \eta_{N_{ts},n} \end{pmatrix} - \mathbf{A}_m \begin{pmatrix} \eta_{1,n-1} \\ \vdots \\ \eta_{N_{ts},n-N} \end{pmatrix}\right) > \boldsymbol{\varepsilon}, \tag{6.15}$$

$$\left(\begin{pmatrix} d_{1,n} \\ \vdots \\ d_{N_{ts},n} \end{pmatrix} - \mathbf{A}_m \begin{pmatrix} d_{1,n-1} \\ \vdots \\ d_{N_{ts},n-N} \end{pmatrix}\right) > \boldsymbol{\varepsilon} - \left(\begin{pmatrix} \eta_{1,n} \\ \vdots \\ \eta_{N_{ts},n} \end{pmatrix} - \mathbf{A}_m \begin{pmatrix} \eta_{1,n-1} \\ \vdots \\ \eta_{N_{ts},n-N} \end{pmatrix}\right).$$

[7]This set of equations is abridged because of space constraints. Please see page 129 for the full matrix definitions.

Representing the left hand side of the equation as $\bar{\mathbf{e}}_m$ and the expression containing noise terms on the right hand side as ζ_m, the equation above can be compactly rewritten as,

$$\bar{\mathbf{e}}_m > \boldsymbol{\varepsilon} + \zeta_m. \tag{6.16}$$

To make the problem convex, the decision variables (model coefficients) are separated from the random terms by using an estimate of the model coefficients. In essence ζ_m is approximated as,

$$\zeta_m = \begin{pmatrix} \eta_{1,n} \\ \vdots \\ \eta_{N_{ts},n} \end{pmatrix} - \mathbf{A}_{m_{est}} \begin{pmatrix} \eta_{1,n-1} \\ \vdots \\ \eta_{N_{ts},n-N} \end{pmatrix}. \tag{6.17}$$

In practice, it is not possible to separate the noise from the data in such a fashion; however approximating the problem in this form is well known to lead to convex formulations with efficient solutions [204]. Using this approximation for both the m^{th} and $m + s^{st}$ model, the resulting reformulation is shown below in problem (6.18).

$$\begin{aligned} &\underset{\mathbf{A_m},\mathbf{A_{m+1}},\boldsymbol{\varepsilon}}{\text{maximize}} \quad \mathbb{P}(|\bar{\mathbf{e}}_m(\tau_m)| - \boldsymbol{\varepsilon} > \zeta_m) \cdot \mathbb{P}(|\bar{\mathbf{e}}_{m+1}(\tau_m)| \leq \boldsymbol{\varepsilon} + \zeta_{m+1}) \\ &\text{subject to} \\ &\qquad \mathbf{e}_m(t) = \mathbf{z}_{t-1} - \mathbf{A}_m \omega_{t-2}, \qquad \text{s.t. } |\mathbf{e}_m(t)| \leq \boldsymbol{\varepsilon},\ \forall t \in [\tau_{m-1}, \tau_m) \\ &\qquad \mathbf{e}_{m+1}(t) = \mathbf{z}_t - \mathbf{A}_{m+1}\omega_{t-1}, \quad \text{s.t. } |\mathbf{e}_{m+1}(t)| \leq \boldsymbol{\varepsilon},\ \forall t \in [\tau_m, \tau_{m+1}) \end{aligned} \tag{6.18}$$

The final approximation of the objective function employs results shown in Chapter 2 of [204]. The results showed that non-convex probabilistic objective functions can be optimized if the probabilities are described by log-concave distributions. To find the solution, the optimization is carried out with respect to the log of the probabilistic objective. The properties of the log function also show that the multiplication of log-concave distributions is also log-concave. Using these results, problem (6.18) is transformed into that of form (6.19).

$$\begin{aligned}
&\underset{\mathbf{A_m},\mathbf{A_{m+1}},\boldsymbol{\varepsilon}}{\text{minimize}} \quad -\,log\left(\mathbb{P}(|\bar{\mathbf{e}}_m(\tau_m)| - \boldsymbol{\varepsilon} > \zeta_m)\right) - log\left(\mathbb{P}(|\bar{\mathbf{e}}_{m+1}|(\tau_m)| \leq \boldsymbol{\varepsilon} + \zeta_{m+1})\right) \\
&\text{subject to} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (6.19) \\
&\qquad \mathbf{e}_m(t) = \mathbf{z}_{t-1} - \mathbf{A}_m\omega_{t-2}, \qquad \text{s.t. } |\mathbf{e}_m(t)| \leq \boldsymbol{\varepsilon},\ \forall t \in [\tau_{m-1}, \tau_m) \\
&\qquad \mathbf{e}_{m+1}(t) = \mathbf{z}_t - \mathbf{A}_{m+1}\omega_{t-1}, \quad \text{s.t. } |\mathbf{e}_{m+1}(t)| \leq \boldsymbol{\varepsilon},\ \forall t \in [\tau_m, \tau_{m+1})
\end{aligned}$$

6.5.5 Finding Preliminary Terms

The approximation presented above in problem (6.19) is still not solvable. The solution of this problem requires an estimate of the distribution of the random vectors ζ_m and ζ_{m+1}, and the estimation of the modeling error bound vector $\boldsymbol{\varepsilon}$. To develop this estimate in this section, recall from Chapter 5 that Φ_{e_m} is the Gaussian cumulative distribution of the modeling errors, σ_m^2 is the variance of the error distribution of the m^{th} model, and ρ is the minimum probability of success used to find deterministic equivalents of probabilistic expressions. In addition from earlier in this chapter, we recall that the number of time series in the multi-attribute representation is N_{ts}, with each time series denoted by γ.

Then to estimate the distribution of the random vectors ζ_m and ζ_{m+1}, here we need an initial estimate of the model coefficients. In this thesis, this estimate is determined by idealizing the problem and assuming independence among the N_{ts} dimensions. Constraining each row, γ, of the vector $\mathbf{e}_m(t)$ to be independent, the probabilities describing a transition point are:

$$\begin{aligned}
\mathbb{P}(|\mathbf{e}_{m_\gamma}(t)| > \boldsymbol{\varepsilon}_\gamma) \quad \forall\gamma = 1 : N_{ts} \\
\mathbb{P}(|\mathbf{e}_{m+1_\gamma}(t)| \leq \boldsymbol{\varepsilon}_\gamma) \quad \forall\gamma = 1 : N_{ts}
\end{aligned} \tag{6.20}$$

In the case of Gaussian distributions, considering the rows independently allows each probabilistic expression to be expressed as a deterministic equivalent [192]. A deterministic equivalent is a linear expression that can be found if there exists a lower bound on the probability ρ such that ρ is at least 1/2.

$$\mathbb{P}(|\mathbf{e}_{m_\gamma}(t)| > \boldsymbol{\varepsilon}_\gamma) \geq p \quad \forall\gamma = 1 : N_{ts}, \quad \rho \geq 1/2, \tag{6.21}$$

$$\mathbb{P}(|\mathbf{e}_{m+1_\gamma}(t)| \leq \boldsymbol{\varepsilon}_\gamma) \geq \rho \quad \forall \gamma = 1 : N_{ts}, \quad \rho \geq 1/2. \tag{6.22}$$

Expression (6.21) can then be reformulated as the deterministic equivalent for an error exceeding the modeling error bound to the positive side,

$$1 - \Phi_{e_m}(\frac{\boldsymbol{\varepsilon}_\gamma - e_{m_\gamma}(t)}{\sigma_m^2}) \geq \rho, \tag{6.23}$$

and a deterministic equivalent for a modeling error exceeding the modeling error bound to the negative side,

$$\Phi_{e_m}(\frac{-\boldsymbol{\varepsilon}_\gamma - e_{m_\gamma}(t)}{\sigma_m^2}) \geq \rho \tag{6.24}$$

Similarly, expression (6.22) can be reformulated as follows,

$$\Phi_{e_{m+1}}(\frac{\boldsymbol{\varepsilon}_\gamma - e_{m+1_\gamma}(t)}{\sigma_{m+1}^2}) - \Phi_{e_{m+1}}(\frac{-\boldsymbol{\varepsilon}_\gamma - e_{m+1_\gamma}(t)}{\sigma_{m+1}^2}) \geq \rho. \tag{6.25}$$

As previously shown in Chapter 5 and the Appendix, in these equations the standard deviation of the modeling error is equal to one plus the squared sum of model coefficients. In addition, the modeling error is defined as the current data point minus the model output. Thus the reformulated equations taken into account all optimization variables: $A_m, A_{m+1}, \boldsymbol{\varepsilon}$.

6.5.6 Convexity of the Solution Space

When solving for the preliminary terms, the solution must be sought over a convex solution space. In the case of N_{ts} dimensions, or time series, the solution space of the problem is made up of $2^{N_{ts}}$ convex subspaces. Thus to search for a solution, the problem has to be evaluated on each of these subspaces, and then the best solution must be chosen out of all possible solutions. In this thesis, the best solution is always chosen to be the solution with the highest probability of success at the transition point.

As an example, consider the case of two dimensions for which there are 2^2 subspaces: ($\mathbf{e}_{m_1} > 0$, $\mathbf{e}_{m_2} > 0$),($\mathbf{e}_{m_1} > 0$, $\mathbf{e}_{m_2} < 0$),($\mathbf{e}_{m_1} < 0$, $\mathbf{e}_{m_2} > 0$),($\mathbf{e}_{m_1} < 0$, $\mathbf{e}_{m_2} < 0$). These subspaces are quadrants that can be depicted in the plane as shown in Fig. 6.2. Then in two dimensions, the problem must be solved four times to determine the number of feasible solutions at a given transition point.

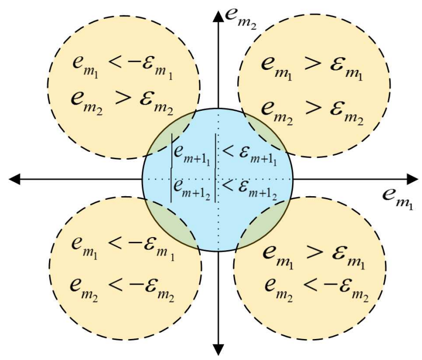

Figure 6.2. Solution Space Illustration for the Case of Two Dimensions (Time Series).

6.5.7 Finding the Approximation in Each Subspace

For each subspace, equations (6.23) - (6.25) are used to formulate a convex problem, the solution of which is an approximation of an initial set of models, $\mathbf{A_m}$ and $\mathbf{A_{m+1}}$, and the modeling error bound vector, $\boldsymbol{\varepsilon}$. In equations (6.23) - (6.25), the symbol Φ_{e_m} is the Gaussian cumulative distribution function (CDF) of the γ^{th} row of the m^{th} MIMO ARX model output. The variances σ_m^2 and σ_{m+1}^2 are determined by finding the squared sum of the coefficients of a given row and multiplying the result by the variance of the observed process noise.

The model determination constraints are then obtained by rearranging equations (6.23) - (6.25). When identifying a set of transition point models with one model output error exceeding the modeling error bound to the positive side, the constraints are,

$$\begin{aligned} \sigma_{m_\gamma}\Phi_{e_m}^{-1}(1-\rho) + e_{m_\gamma}(\tau_m) &\geq \boldsymbol{\varepsilon}_\gamma \\ \sigma_{m+1_\gamma}\Phi_{e_{m+1}}^{-1}(\frac{\rho}{2}+\frac{1}{2}) + e_{m+1_\gamma}(\tau_m) &< \boldsymbol{\varepsilon}_\gamma. \end{aligned} \tag{6.26}$$

Then, when identifying a set of transition point models with one model output error exceeding the modeling error bound to the negative side the constraints are,

$$\sigma_{m_\gamma}\Phi_{e_m}^{-1}(\rho) + e_{m_\gamma}(\tau_m) \leq -\boldsymbol{\varepsilon}_\gamma$$

$$\sigma_{m+1_\gamma}\Phi^{-1}_{e_{m+1}}(\frac{\rho}{2}+\frac{1}{2})+e_{m+1_\gamma}(\tau_m)<\boldsymbol{\varepsilon}_\gamma. \tag{6.27}$$

Because there are N_{ts} time series (dimensions), there are $2^{N_{ts}}$ possible deterministic problems. One formulation showing the case where all modeling errors of the m^{th} model exceed the modeling error bound to the positive side is shown here. The remaining formulations can be similarly derived by the reader.

$$\begin{aligned}
&\underset{\mathbf{A_m},\mathbf{A_{m+1}},\boldsymbol{\varepsilon}}{\text{minimize}} \quad \|\boldsymbol{\varepsilon}\|_2\\
&\text{subject to}\\
&\qquad \mathit{Absolute\ constraints}:\\
&\qquad e_{m_\gamma}>\boldsymbol{\varepsilon}_\gamma \qquad\qquad\qquad\qquad\qquad \forall\gamma=1:N_{ts}\\
&\qquad \mathit{Probabilistic\ constraints}:\\
&\qquad \sigma_{m_\gamma}\Phi^{-1}_{e_m}(1-\rho)+e_{m_\gamma}(\tau_m)\geq\boldsymbol{\varepsilon}_\gamma \qquad \forall\gamma=1:N_{ts}\\
&\qquad \sigma_{m+1_\gamma}\Phi^{-1}_{e_{m+1}}(\frac{\rho}{2}+\frac{1}{2})+e_{m+1_\gamma}(\tau_m)\leq\boldsymbol{\varepsilon}_\gamma \quad \forall\gamma=1:N_{ts}\\
&\qquad \mathit{Model\ fit\ constraints}:\\
&\qquad \mathbf{e}_m(t)=\mathbf{z}_{t-1}-\mathbf{A}_m\omega_{t-2}, \qquad \text{s.t. } |\mathbf{e}_m(t)|\leq\boldsymbol{\varepsilon},\ \forall t\in[\tau_{m-1},\tau_m)\\
&\qquad \mathbf{e}_{m+1}(t)=\mathbf{z}_{t}-\mathbf{A}_{m+1}\omega_{t-1}, \quad \text{s.t. } |\mathbf{e}_{m+1}(t)|\leq\boldsymbol{\varepsilon},\ \forall t\in[\tau_m,\tau_{m+1})
\end{aligned} \tag{6.28}$$

For each possible transition point, a solution is sought for all possible subspaces. A solution to any one of these problems is an estimate of the models $\mathbf{A_m}$ and $\mathbf{A_{m+1}}$ at a particular candidate transition point. The set of identified models is then used to obtain estimates for the covariance matrices Σ_m and Σ_{m+1} of the random vectors ζ_m and ζ_{m+1} using equation (6.8). This estimate is fixed for the remainder of the solution, and the models $\mathbf{A_m}$ and $\mathbf{A_{m+1}}$ are optimized using problem (6.19) and starting from the initial set.

6.5.8 Implementing the Approximation

Having completed the approximation of the problem it is now useful to summarize the approach that will be used at each candidate transition point. The procedure is summarized below in table 6.2. The first step is to choose the candidate transition point. Then for each of the M subspaces, we attempt to find a solution to the idealized problem of type (6.28). If for any problem a solution is found, then the initial set of models $\mathbf{A_m}$ and $\mathbf{A_{m+1}}$ is used to find and fix the covariance matrices Σ_m and Σ_{m+1} using equation (6.8). Then the solution set of models is tuned using problem (6.19). The final transition point chosen is the transition point that maximizes the probabilities described in problem (6.13).

Approximation Procedure for Problem (6.13)
1. Choose a candidate transition point from the data.
2. Try to find a solution to each of the N_{ts} problems of form (6.28).
3. For each solution found, determine Σ_m and Σ_{m+1} using equation (6.8).
4. Approximate the solution to problem (6.13) by problem (6.19).
5. Choose the transition point with the highest probability of success.

Table 6.2. Approximation Procedure When Representing Multiple Time Series

Because the exact solution to the problem is computationally expensive, and the computational resources are usually limited, an approximation to the problem is needed to enable the representation of data using linear models. Furthermore, this approximation serves to make the problem scalable such that new data can be added to a pre-built reference map.

6.6 Data Representation Extraction Algorithm

The approximation procedure described in table 6.2 is part of an iterative process that creates a data structure to be used for in-sequence localization. The full process is described below in table 6.3. Categorizing the approach in terms of data representation approaches, the algorithm presented here is a top down, sliding window approach. This means that data segmentation is performed starting from a single segment and ending with the largest number of segments below. The segmentation can be considered a sliding window method because every possible segmentation end point is tested before the most robust point is chosen as the segmentation point. The following is a more precise description of the data structure/representation algorithm that is followed by a summarized algorithm in table 6.3.

Data Segmentation: When beginning to segment a data set, $\mathbf{D}$, the start and end points are defined as k_s and k_e, respectively. These points can also be adjusted if only a subset of the data is to be segmented. Once k_s and k_e are set, two initial segments are formed: the smallest possible segment starting at k_s and the rest of the data. The size of the smallest segment in this chapter was chosen to be $2N$. The end point of the first segment is a candidate transition point, t, about which the approximations described in the preceding section are attempted. There are $2^{N_{ts}}$ approximations that are attempted. For each successful optimization solution, the feasible solutions are saved. The saved data includes the models A_1, A_2, the transition point, τ_1, the modeling error bound vector, $\boldsymbol{\varepsilon}$, and the probabilities of success for each model.

Next one data point is transferred from the longer data segment to the shorter data segment. The new candidate transition point is $t+1$, and the 2^N_{ts} optimizations are again performed, seeking any solutions to the problem. The solutions are then saved at candidate transition point τ_2. This process is iterated until all possible transition points have been evaluated. Then from among all possible transition points, the point chosen is that which maximizes the probabilities shown in problem (6.13).

Having chosen the most robust transition point, the resulting segments are recorded in the data structure. Because the first level of the structure is a model that describes all data (used to calibrate the $\boldsymbol{\varepsilon}$ value), the first partition is recorded at the second structure level, $L = 2$. At the next level, the procedure is iterated, but this time a transition point is found in each of the two segments determined at $L = 2$. The models found have tighter $\boldsymbol{\varepsilon}$ bounds, and the shift in dynamics is usually smaller than in the previous level. The data segmentation continues until no more transition points can be found that satisfy the minimum probability of error p.

Data Tiering: To fully take advantage of the properties of Gaussian noise, the data is also tiered by averaging the observed data by orders of magnitude. Averaging the Gaussian noise reduces its standard deviation by the square root of the number of averages. Thus for in-sequence localization, averaging the data reduces the uncertainty caused by additive sensor noise and correspondingly it increases the certainty of the location estimate.

Data representation on tiered data is performed starting with the largest data decision. When all possible transition points have been identified for a given tier, the next tier is started. Segmentation is performed on the segments from the last level of the previous tier's model structure. The new, finer data decimation introduces new dynamics on which to segment and a finer data decimation on which to localize. When all tiers are segmented, the final data structure has L levels with M_L segments per level.

Reference Map Creation Algorithm
Initialization
$L = 0$
$\tau_0 = 1$
$\tau_1 = N_{ML}$
$tierflag = 0$
σ_η^2 = to expected noise level
FOR tier = 1:Max tier
obtain data at the tier resolution
WHILE $tierflag == 0$
$L = L + 1$
FOR m = 1:N_{ML}
$n = 0$
$k_0 = \tau_{m-1}$ + N
$k_e = \tau_m$
$\tau_n = k_0$
FOR t = k_0:k_e
Attempt each of the convex approximations
IF a solution is found
record t, A_m, A_{m+1} in a structure S
iterate the transition point counter: n = n + 1
END FOR
IF n == 0 && $m == M$
$tierflag = 1$
ELSEIF $m < M$ && n > 0
add n to the total number of transitions on the level
from S choose τ_n s.t. τ_n maximizes:
$\mathbb{P}(\lvert\mathbf{e}_m\rvert\,(\tau_m)_\gamma > \varepsilon_\gamma) \cdot \mathbb{P}(\lvert\mathbf{e}_{m+1}\rvert\,(\tau_m)_\gamma > \varepsilon_\gamma)$
record the new segments on L
END IF
END WHILE
END FOR

Table 6.3. Data Structure Extraction Algorithm for Multiple Time Series.

6.7 In-Sequence Localization Algorithm

The data structure containing MIMO ARX model representations of segments of data is used now for in-sequence localization. In practice, in-sequence localization is the sequential elimination of possible data locations until a single viable location remains. The model-based representation described in this chapter is specifically tailored to this application. In particular, at first, a small set of data points (corresponding to the length of a data model) is collected. This data is stored in matrices called x_{tp} and X. The data is then sequentially input into each model in the data structure starting with model 1 on level 1. Only models whose parent models[8] are feasible are evaluated.

Model feasibility is evaluated by bounding the noise observed on each data point as $|\eta_k| \leq \eta_B$[9] and attempting to find a set of constants $\bar{\eta} = [\eta_{k-1}, ...\eta_{k-N-1}]$ such that the resulting modeling error is smaller than the modeling error bound ε, as described in equation (6.2). Models whose errors fall outside of the modeling error bound are considered infeasible, and models whose errors fall inside of the modeling error bound are feasible.

The feasible models are saved in a tree structure such that each possible vehicle path has its own branch leading to a single leaf at the bottom of the tree, termed the result tree.. After each consecutive data point is acquired, the result tree is reevaluated and a new tree is re-created. Thus the result tree is flexible and is continuously reduced by the elimination of models. The localization algorithm ends when a single feasible model remains at the bottom level of the result tree. This in-sequence localization algorithm is described below in table 6.4.

6.8 Numerical Experiments

The MIMO ARX data representation trees that are proposed here can be used to identify a fragment of data acquired after a complete process has been recorded. This section demonstrates that the proposed approach can work on a variety of

[8]Parent refers to the data segment on the previous level from which the current segment is extracted.

[9]Here η_B is twice the standard deviation of the expected Gaussian noise. This standard deviation is typically provided by the sensor manufacturer.

diverse data sets. Three different data sets are used in this section such that possible applications of this work are presented and the efficiency of the developed algorithms is demonstrated.

The first subsection, section 6.8.1, shows numerical experiments that validate the main motivation of this thesis, vehicle localization. In these experiments, the time series of vehicle pitch, representing road grade, and vehicle roll, representing the cross-sectional profile of the road, are used for localization. The localization results are displayed and the results format is described to aid the remaining problem descriptions.

The second subsection, section 6.8.2, presents numerical simulations using wind speed and solar irradiation data that are collected at the National Renewable Energy Lab in Golden, CO USA. This data is used to facilitate variable generation forecasting [194], which is the estimation of power generation capabilities of renewable energy sources. The goal of this section is not to present a method of variable generation forecasting, but to propose a method of selection of the dominant renewable generation source such that generation strategies can be optimized in sequence of greatest potential instead of via complex local models. This section introduces the problem to be solved and shows that in-sequence localization is a good candidate for the identification of power generation patterns in data.

The last subsection, section 6.8.3, demonstrates via numerical simulations the applicability of the in-sequence localization algorithms in the environmental domain. In particular, we focus on the problem of stream temperature monitoring. This is a significant problem in the western parts of the United States, where sophisticated water management methods are used to supply water to arid regions. This section introduces the problem and then develops a reference map and localization experiments on the available time series of stream flow and air temperature. These experiments demonstrate the feasibility of using in-sequence localization to estimate an upper bound on the stream temperature.

6.8.1 Vehicle Localization

This section provides the numerical experiments to show that the proposed method is an effective vehicle localization method and to verify that the experiments are correctly coded with respect to the previously described work in chapters 4 and 5. The vehicle localization results are shown below in Fig. 6.3. To generate this figure, a simulated vehicle localization experiment was performed at each possible starting point on the map, and the localization data was corrupted with 12 dB of White Gaussian noise, which matches the work presented in chapters 3-5.

The results in Fig. 6.3 demonstrate the outcomes of these experiments. The first plot in the top left of the figure shows a histogram of the wall clock time for each experiment. In other words, this plot shows the computation time in seconds of each localization experiment. We note that most localization experiments ended by finding the correct location within 50 seconds. The middle subplot in the top row of the figure shows the localization distance in terms of the steps taken during the localization process. In this case, each step was 0.25 m, resulting in a maximum localization of 250 m. We note here that the majority of the experiments were completed in half this distance. The final subplot in the top row of the figure shows the distance of the final localization point from the nearest transition point. This subplot shows the significance of transition points in localization. Values of distance close to 0 m signify important transition points that lead to rapid localization. Values that are spread further from the origin signify less important transition points.

The final subplot in the bottom of the figure shows the IMU data time series on which the localization experiments were performed. The pitch time series is represented with the dashed line, and the roll time series is represented with the solid line. The identified transition points are shown as vertical lines that intersect the time series at a given distance from the first point in the time series. This distance is expressed in meters. Overall, these experiments show that localization was rapid, efficient, and accurate. In these experiments, accurate was defined to be within 100 m of the correct location, but the practical convergence was often much closer. The comparison of these results to previous single time series localization results will be further discussed below in section 6.9.1.

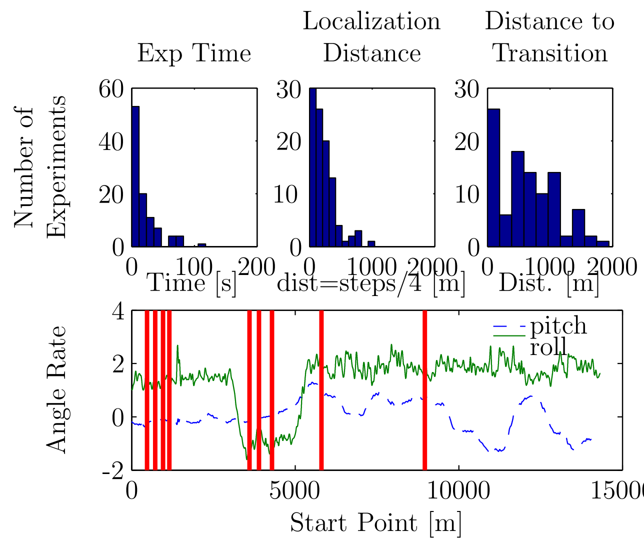

Figure 6.3. Example of Localization Using Multiple Time Series of Vehicle Data.

6.8.2 Dominant Source Localization in Power Generation

The inclusion of multiple new and renewable sources introduces instability in the power grid [194]. This instability is inherent in wind and solar energy because this to energy being harvested from natural, cyclical phenomena[10] that occur in a somewhat unpredictable fashions. This instability creates a new challenge for power grid managers who need to dynamically alter the generation mix using conventional sources that may switch on and off at rates slower than the change rate in the power derived from the renewable source.

To address this problem, researchers have sought to develop models that predict (forecast) the power production of each renewable source. Examples of this include models for solar irradiation and wind power [193, 205]. In each case, the generated power is predicted from the change in the underlying phenomenon (wind speed, solar irradiation, or cloud cover) because the power output is proportional to the environmental constants of instantaneous solar irradiation and wind speed.

For wind speed, two different types of forecasting models are used. The first is a meteorological model that predicts long-term trends in the data, and the second is a short-term model (typically linear) that predicts the near term[11] switching of wind power [206, 207]. In solar data numerical models termed Numerical Weather Prediction are necessary because the key predictor of solar power is cloud cover

[10]yearly wind patterns, solar patterns, cloud cover

[11]This is typically in the 1-12 hour time horizon.

[208].

The use of wind speed data is particularly complex because it has both deterministic components and random, non-stationary, and non-Gaussian components. Seeking to mitigate the effects of non-stationary and non-Gaussian components, researchers have often stationarized the data [207] and used a power transform returning the data to a Gaussian form [209]. The number of models generated in this manner has been large because each model must be tuned to the local prediction application [194].

The large number of publications that tune both wind speed and solar irradiation prediction models for specific sites suggests that the current approach in forecasting is too local to create a cohesive theory of electric source integration. This is generally problematic for two reasons. First, changing climate conditions may require a re-calibration of models without adequate historical data. Second, grid operation has been historically performed by skilled operators whose intuition filled the gaps in generation understanding. Many local and imperfect models add confusion in the grid operation and therefore introduce further instability.

Exploiting the in-sequence localization approach, one possible alternative is to rank the energy sources according to their power generation potential on any given day. This would enable a clearer approach in optimizing the generation source mixture. The main advantage here is the decoupling of cyclical (seasonal) patterns in the solar and wind data from the stochastic elements that make forecasting difficult.

To be clear, this limited example does not aim to show that these algorithms outperform previous forecasting approaches, but rather shows an application of in-sequence localization, in which dominant power sources can be determined and previously used models can be used to forecast the precise power generation potential. For this demonstration, a four year set of solar irradiation and wind speed data is used. This data set obtained from the National Renewable Energy Lab[12] in Golden, CO USA is run through a ten-day moving average to smooth the stochastic effects in the data.

Fig. 6.4 demonstrates a series of in-sequence localization experiments that are performed on the time series of data. The top left hand plot shows that in-sequence

[12] $http://www.nrel.gov/midc/nwtc_m2/$

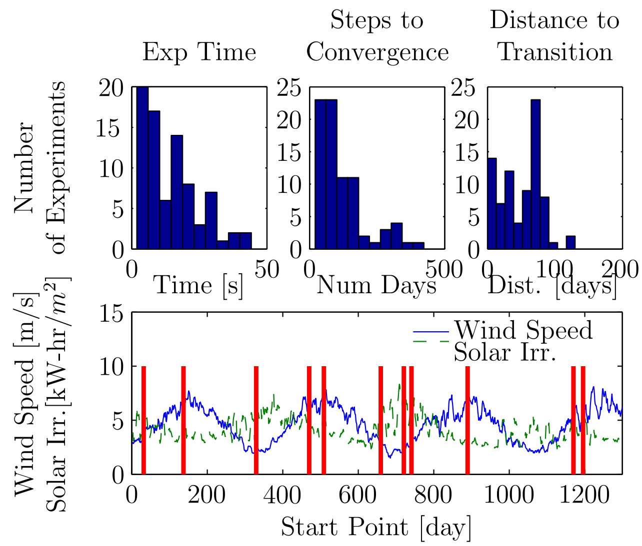

Figure 6.4. Example of In-Localization in Wind Speed Data and Solar Irradiation Data.

localization was computationally efficient, completing the task in less than 50 s of wall clock time for each performed experiment. On average the number of days of data needed to identify the seasonal location in the data is 250, reflecting the observed spacing of the transition points in the reference map. The significance of these points can be seen in the top right hand plot, where localization is seen to occur within 100 days of a transition point. The bottom plot shows the wind speed data and the solar irradiation data with the identified transitions throughout.

6.8.3 Localization Using Air Temperature and River Discharge

Another application that can benefit from in-sequence localization is environmental monitoring. Monitoring stream temperature is necessary because the diversion of water from regions with a higher level of water supply to arid regions in the western United States leads to reduction in the water levels of streams and rivers on the supply side. Streams and rivers with lowered water depths have a decreased thermal buffering capacity and are therefore at risk for overheating, killing off cold water fish species that are the cornerstone of their ecosystems [210].

Unfortunately stream temperature is a quantity that is rarely measured because of a lack of sensors in the field. Instead, water managers have focused on using aerial images and image recognition techniques to estimate the water temperature [210, 211]. The drawback of this is two-fold. First the resolution in first and second order streams[13] is poor [212], leading to false estimates, and second, aerial images suffer from occlusions by tree covers similar to blockages of the GPS signal by trees and tall buildings. To remedy these problems, several authors have proposed an estimation of the stream temperature using models (often linear) and available inputs such as stream flow and air temperature, which are commonly available [197, 196]. Special interest is paid to approaches that are computationally efficient such that water managers can incorporate them into online planning tools that can be expeditiously used to make adjustments in the fields, i.e. the release of water from reservoirs to impact downstream depth [195, 213].

In this thesis, we do not propose a method to estimate the stream temperature. Instead, we take note of two facts observed in the prior publications. First, the calculable quantity of interest is not the per minute stream temperature but rather the upper bound on the estimate of the maximum daily stream temperature. Second, the previously published approaches [210, 211, 195, 213] show that at least some streams have recorded flow, air temperature and water temperature data.

Therefore, if all three time series (flow, air temperature, water temperature) are available for a given depth in a stream little or no tree cover, then the maximum[14] stream temperature can be indexed by the flow and air temperature. To show that this indexing is possible, we use data from the gauges at the Gordon Gluch Stream near Boulder, CO. that is part of the Boulder Critical Zone Observatory[15]. Stream flow and air temperature data are smoothed using a 24-hour moving average to reduce the effects of sensor noise in the data. The data are indexed in a reference map, and the approximate location of the stream in the cycle is identified for a random start point. The results of the localization experiments are shown in Fig. 6.5.

In the top of Fig. 6.5, the results show that computationally, the in-sequence localization was efficient, taking less than 50 seconds to complete. This is a highly desirable outcome for real-time stream temperature monitoring implementation

[13]First and second order streams are at the top of the stream network, with no tributaries, i.e. these are streams formed by primary mechanisms of water runoff - rain and snow melt.

[14]Maximum stream temperature is the result of full solar irradiation in the absence of shade.

[15]$http://czo.colorado.edu/query/GGU_SW_0.shtml$

because numerous models are simultaneously running to monitor many aspects of the water system. In addition, less than 300 hours (12 days) of stream data is needed to locate the current stream state. This shows that the approach minimizes the number of data points stored in between monitoring sessions. As a comparison, this means that less than 7% of the historical record needs to be sampled in order to estimate the current flow and temperature state.

The bottom plot of the figure shows the water flow (discharge) and the associated air temperature near the Gordon Gulch Stream. The air temperature time series is represented with dashed lines. The identified transition points are shown as vertical lines. Note that the transition points clearly capture hydrologically significant events such as the “falling limb” of stream flow before a subsequent peak and a drop in temperature.

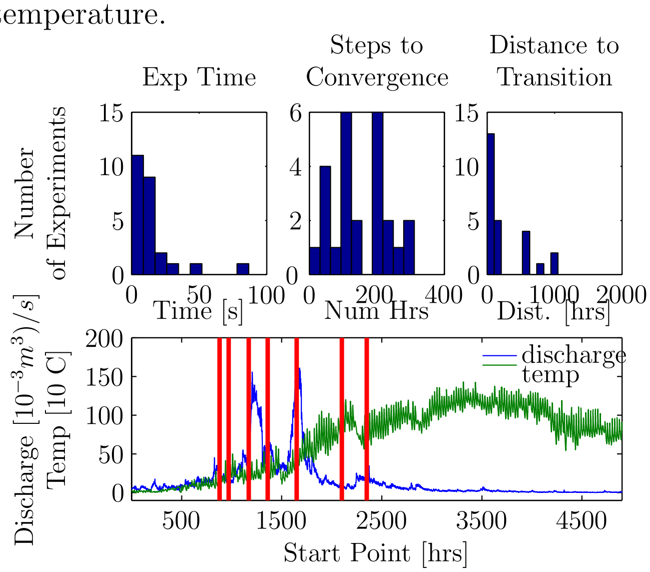

Figure 6.5. An Example of In-Localization For Stream Health Monitoring.

6.9 Discussion

6.9.1 Contribution of Additional Time Series

A pivotal question in this chapter is whether the introduction of additional dimensions adds to the in-sequence localization performance. This is particularly important because the reference map is enlarged, and correspondingly the computational cost of localization is increased. In this study, it was found that localization does benefit from the addition of multiple time series to the reference map. This finding is illustrated below in Fig. 6.6 and Fig. 6.7. The first figure, Fig. 6.6, shows the segmentation and localization results using the probabilistic method developed in [24]. These in-sequence localization experiments are performed on the vehicle pitch data. The second figure, Fig. 6.7, is a repeat of the localization figure shown above for the vehicle experiments.

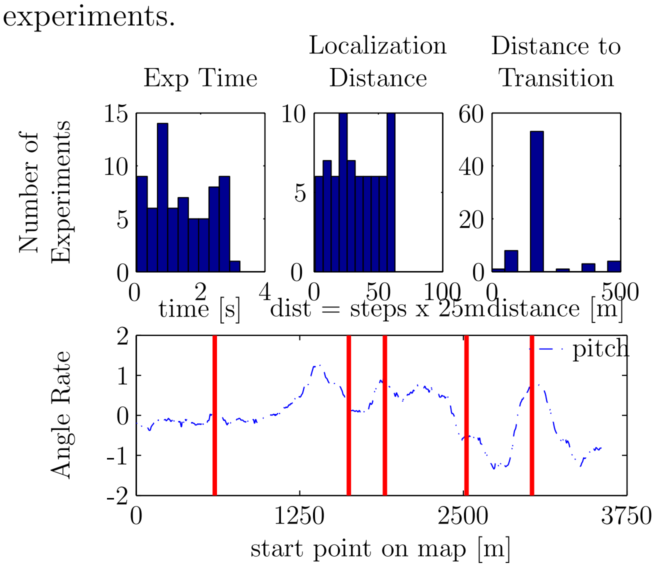

Figure 6.6. Segmentation and Convergence using Only Vehicle Pitch Data.

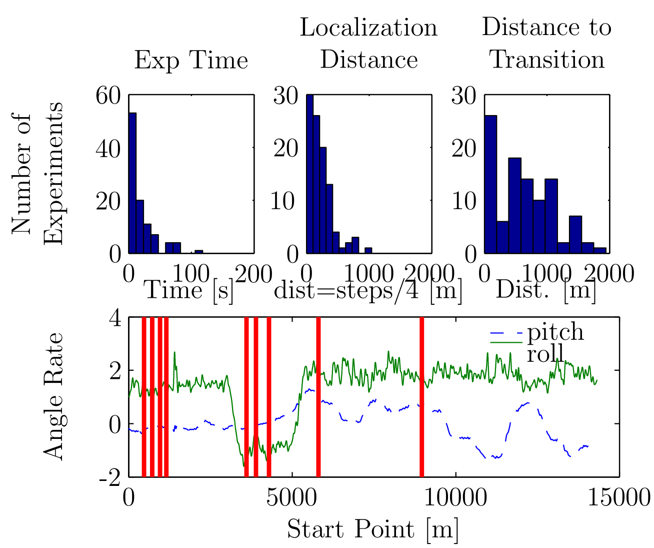

Figure 6.7. Vehicle Segmentation and Convergence Results Using Both Vehicle Pitch and Roll Data.

These figures illustrate the advantages and disadvantages of multiple time series in a reference map. On one hand, the experiment time is clearly longer by at least an order of magnitude. On the other hand, the localization distance is drastically reduced by a factor of 3, and the significance of individual transition points is dramatically reduced. Furthermore, new transition points are found in areas of the time series that were previously constrained to a single segment, such as, in the interval of distances 0 m to 1250 m. Thus these figures show that localization is improved and the map reliability is improved by the addition of time series. The cost of this is the increased computational power required for localization.

It is important to note that these time series clearly contributed information to the map. The new information introduced in the map is represented by the newly determined transition point. From a theoretical point of view, there is no *a-priori* approach to determining whether a time series will contribute to the in-sequence localization map. Instead the multi-attribute representation must be determined, and the contribution of the new time series can then be evaluated. From a practical point of view, it is often possible to identify new information by considering the placement of sensors in space, e.g. the vehicle pitch and roll which are inertial measurements in orthogonal directions. Lastly, note that adding time series with redundant information yields no benefit and only increases the computational cost without improving the in-sequence localization results.

6.9.2 Limitations on Adding Time Series

The main limitation of this work with respect to additional time series is the need for correlation among the time series. In this chapter it is assumed that the time series are correlated and can be considered “attributes” of an underlying process. For example, in the case of vehicle localization using inertial measurements, the time series of pitch measurements and roll measurements are used to demonstrate the creation of localization maps and the in-sequence localization process. Additional correlated time series could also help the localization process. In contrast, adding time series that are not correlated leads to the inability generate a reference map.

For example in vehicle data, in addition to pitch and roll measurements, a third axis of the IMU exists. This axis is called yaw and it describes the left to right motion of the vehicle (eg. turns). One can assume that the time series describing vehicle yaw is uncorrelated or weakly correlated to the road profile since an arbitrary number of turns can be taken by a driver and these turns do not have to be related to the road surface geometry. Indeed, additional experiments performed for this thesis tested the inclusion of the yaw measurements during the reference map building. The results showed that *no transition points* could be found across all three time series: pitch, roll and yaw; and therefore, no reference map could be built.

This shows that the yaw measurements did not contribute meaningful localization information. On the other hand, these three time series could be used in a reference map if the relaxed problem in (6.11) is used. This formulation would allow the identification of the transition points on the pitch, roll, and yaw time series to be identified independently. However, the solution to relax the initial problem formulation could lead to the problem that an arbitrary stop by the mapping vehicle would lead to a reference map that is not useful to subsequent drivers.

6.9.3 Reference Map Creation Without Time Series Normalization

In the beginning of this chapter when the assumptions in this algorithm were stated, equation (6.8) was shown with a standardized (identical) noise variance for each time series. This standardization of the noise variances enabled the identification of transition points across all time series. If the noise variance is not standardized, the resulting segmentation will find the transition points associated with the less noisy time series. For example, using vehicle data, the standard deviation of the noise in the vehicle pitch time series was set to be an order of magnitude greater than the standard deviation of the noise in the vehicle roll time series. The segmentation procedure was executed and a reference map was created. This reference map is visualized below in Fig. 6.8.

Note that in contrast to the transition point map shown in Fig. 6.7, the map here shows only two transition points that corresponds to the changes seen in the vehicle roll time series. Thus, as expected, the time series with the smaller noise variance was used as the primary reference. This brief experiment shows the necessity of normalizing all time series such that their noise standard deviation is

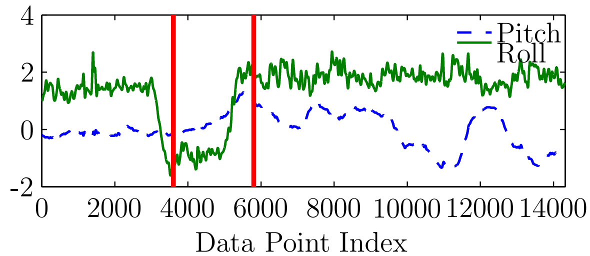

Figure 6.8. Segmentation Results With Different Noise Variances.

equal.

6.10 Conclusions and Future Development

This chapter presented an approach to incorporating multiple (attribute) time series into a data representation. The approach builds on the chance constrained framework used in chapter 5 and allows even greater flexibility in defining the stochastic disturbance. The multiple attribute time series are represented using multi-input multi-output models. Each model represents an interval of data and the robustness of the models to process noise is optimized about the transition points between the models. The resulting solutions to the chance constrained program and to the larger in-sequence localization problem are tested in three separate applications: vehicle localization, renewable energy generation, and environmental health monitoring. These applications showed the versatility of the approach and new avenues of research following the completion of this thesis.

```
In-Sequence Localization Algorithm
==================================
Initialization
----------------------------------
Collect N + 1 data points:
set loop index: k = N + 1
initialize data matrices: x_{T_j} and X_j
----------------------------------
Begin Localization Loop
----------------------------------
FOR l = 1:L
  FOR m = 1:M_L − 1
    IF the parent segment ofA_m is feasible:
       Find a set η̄_m s.t. |e_{m,k}| ≤ ε_m
       Find a set η̄_{m+1} s.t. |e_{m+1,k}| ≤ ε_{m+1}
      IF η̄_m exists
        A_m is feasible, add to its counter
        add the m^{th} segment to the result tree
      ELSEIF η̄_m does not exist AND η̄_{m+1} exists
        then the data point d_k is a transition point
        add the m^{th} segment to the result tree in case of error,
        add the m + 1^{st} segment to the result tree,
      ELSE
        remove the m^{th} segment from the result tree
      END IF
    END IF
  END FOR
END FOR
Collect an additional data point
iterate the index: k = k + 1
iterate the data matrices: x_{T_j} and X_j
----------------------------------
Following Each Detected Transition Point
----------------------------------
FOR l = 1:L
  FOR m = 1:M_L
    IF (τ_m belongs in the segment &&
        segment m is not in the result tree)
       add the m^{th} segment to the result tree
    END IF
  END FOR
END FOR
```

Table 6.4. In-Sequence Localization Algorithm for Model Structures Using MIMO ARX Models.

Chapter 7

Concluding Remarks

Concluding this thesis, this chapter provides a concise overview of the material that has been presented in the form of contributions that have been made to the scientific literature. We review the contributions that each chapter has made with respect to previously published work in the literature and to the previous chapters in the dissertation. Lastly, the final section of this chapter discusses future work that can grow out of this dissertation and that will enable even more realistic applications of the problem of in-sequence localization.

7.1 Contributions

7.1.1 The Introduction of In-sequence Localization

The opening (third) chapter of this dissertation begins with the introduction of the problem of in-sequence localization. This is a new problem that is made possible by the ability to collect large data sets. In-sequence localization is the location of the most recently collected data point inside of a previously recorded time series.

To provide a context for the development of the dissertation, this chapter explores seven previously published data representations that are feasible candidates for in-sequence localization. These representations are the Piecewise Aggregate Approximation [28, 29], the Discrete Wavelet Transform Representation [161], the Symbolic Aggregate Approximation [54], the Discrete Fourier Transform [47], the Chebyshev Polynomial Representation [162], the Piecewise Linear Representation

[56], and the Adaptive Piecewise Constant Approximation [163]. Each method is described in the chapter and an in-sequence localization algorithm is presented for fixed data window representations and adaptive data window representations.

We note that a primary factor in the performance of in-sequence localization algorithms is the sensor noise that is added during data collection. For this reason, the representations are tested both for fidelity in representation in the presence of noise and for in-sequence localization performance. Four different data sets are used for the testing: two vehicle data sets and two synthetic data sets. The chapter concludes by hinting at two possible approaches to creating dimension reducing representations for in-sequence localization. The contributions in this chapter are the introduction of the problem of in-sequence localization, the survey of available representations, and the detailed testing of the published data representations with respect to both sensor noise and the in-sequence localization problem. The results presented in this chapter are under preparation for submission to IEEE Transactions on Knowledge and Data Engineering.

7.1.2 An ARX Model-based Approach to In-Sequence Localization

The forth chapter introduces ARX models as data representations that address some of the weaknesses of previously published data representations in the context of in-sequence localization. In essence, using the models eliminates the need for the computation of a model agreement metric and instead compares each model output to its modeling error bound. Agreement or disagreement with this bound determines the segment feasibility. Using linear models allows the data acquisition to begin in the middle of a data segment and the model parameters to be tuned to counteract the effects of noise. In addition the models are also structured in a tree-like fashion to reduce the computational burden of the online in-sequence localization process. Thus this chapter shows that detailed linear models are advantageous in reducing the complexity of both the mapping and the localization mechanisms for in-sequence localization.

In this chapter, the testing of the developed algorithms is performed on vehicle terrain data and is focused on the application of vehicle localization. However, the algorithms can also be used in fields such as data mining where large databases need to be efficiently represented for subsequence identification. In contrast to authors who have also used linear models in data management [122, 93, 53], this chapter uses models of greater dimension that retain more detail from the data. These models of greater fidelity eliminate the need for a matching metric during the localization process, reducing the computational complexity and algorithm design uncertainty. Preliminary work in this chapter appears in the proceedings of the 2012 IEEE Conference on Decision and Control. The work presented in the chapter is currently in press in IEEE Transactions on Intelligent Transportation Systems.

7.1.3 A General Approach to Robust ARX Data Representations

A key aspect in data representation and time series analysis is noise. However, noise is frequently ignored when formulating a data representation. The contribution of the fifth chapter of this thesis is to present one approach to building robust data representations that take into account two types of noise. The first type of noise is a deterministic noise that is observed due to interference between devices [185] or vibrations in the recording sensors [186], and the second type of noise is stochastic sensor noise that is observed in any practical data collection device. The consideration of both types of noise is critical in data representation [187].

The deterministic noise considered in this chapter is sinusoidal noise. Sinusoidal noise, which occurs in IMU data [186] and is of interest in many other applications including speech processing [185], is particularly destructive to data representations because strong cyclic noise components can obscure the data points about which representations are determined. To address this problem, the internal model principle is used to remove sinusoids of a known frequency automatically during the in-sequence localization. Stochastic noise can be equally destructive. In particular in IMU data, the effects of the noise are accentuated when the data is integrated to obtain quantities such as position and velocity [146]. In this chapter, the data representations are tuned to automatically reduce the effects of stochastic noise, provided that the distribution of the sensor noise is known.

Our approach is to choose robust models that mitigate the effects of both deterministic and stochastic noise such that the overall computational burden of in-sequence localization is reduced by increasing the speed at which erroneous paths are eliminated. In addition, in this chapter the localization process itself is optimized. An in-sequence localization process that terminates itself after a user defined accuracy is presented. This localization mechanism, which is based on the robust data representations, demonstrates the effectiveness of the representations for in-sequence localization. In addition, when viewing the results of localization in parallel with the representation map, the adaptive approach to in-sequence localization reveals the strengths and weaknesses of the data sensor, which leads to the logical method of evaluating the addition of sensors with respect to the amount of information they contribute to in-sequence localization.

Lastly, this chapter introduces a new domain for the problem of in-sequence localization: financial data. Financial data provides an interesting domain for in-sequence localization because it is readily available and finely sampled. In particular, this chapter shows that when using US inflation data the robust segmentation points that are determined are points in time that delineate the beginning and end of major US recessions and depressions. The algorithms are also applied to random data to demonstrate that any time series can be segmented using our algorithms.

7.1.4 A Robust Multiple-Time Series Representation Method

The final chapter in this dissertation contributes to the body of literature by expanding the developed data representations to multiple dimensions, i.e. including an arbitrary number of time series in the data representation and in-sequence localization algorithms. Multi-Input Multi-Output ARX models are used to simultaneously describe each collected time series, where each series represents data from a new sensor. Each time series introduces a new dimension in the representation, and thus the multiple time series representation in this chapter is a multidimensional time series representation.

In general, the formulation of the problem in multiple dimensions results in a highly non-convex problem. While solvable, the solutions to the non-convex problem require considerable computational time and do not accommodate the iterative search for robust points in the time series. This chapter presents a convex approximation to the problem that can be rapidly solved for an increasing time series size.

In addition to introducing this problem to simultaneous representation of multiple time series, this chapter also extends the use of the MIMO ARX data representation into the fields of environmental science and energy production. In the first case, this chapter shows that the algorithms presented in this dissertation can be used to reference previously stored stream flow data and air temperature data (over the same stream) with the goal of inferring the stream water temperature, which is a typically unknown quantity. In the second case, we show an indexing of solar irradiation data with wind speed data. The indexing of this historical data can be used in variable [renewable energy] generation forecasting to predict the switching points in time when solar energy is assigned a higher load percentage vs. when wind energy is assigned a higher load percentage. These switching points are critical in a dynamical generation system because other slower generators may need to be ramped up or down to accommodate the variability in the renewable resources.

7.2 Future Work

While the research in the final chapter of this dissertation ends one stage in the development of the application of in-sequence localization, there are many further avenues of research that one could take in developing this work. In particular, the work presented in the dissertation follows the development of a data representation on the map-making side of localization. Yet the optimization of the online localization aspect of the in-sequence localization is also a promising area of research.

7.2.1 Robust Model Trees

The development of the research presented in this dissertation used a bisection tree, where each consecutive level bisected the data intervals from the previous level. A natural end to the segmentation process was found by implementing a probabilistic minimum of success about the model transition point in the presence of sensor noise. However, no evidence exists that this bisection tree is the best method of representation storage. In fact, there are numerous types of trees that have been developed for database applications that may improve the robustness of the developed algorithms [214]. Studying the available methods with respect to in-sequence localization may yield a more suitable model structure for localization or may suggest a new model structure to improve the current in-sequence localization approach.

7.2.2 Simultaneous Optimization Across All Models on a Tree Level

In addition to trees that have been specifically developed for noise robustness, a larger optimization problem can be pursued that simultaneously optimizes all models on a tree level. The optimization would be such that the models have a maximum probability of success at their respective transitions and at most a minimum probability of success at other transition points. This problem is highly non-convex and the number of terms in the objective grows rapidly with increasing model numbers, but an appropriate approximation would be of great interest to the data mining community because the identified models would automatically determine the most mutually unique set of models.

7.2.3 Robust Path Selection Methods

On the localization side of the problem, the in-sequence localization speed can be dramatically increased with the advent of a path selection tool. In this dissertation, localization was performed until a single viable path was left surviving. However, there are clearly more metrics about particular applications (ex. sensor placement, sequential nature of data, etc.) and about the model segments themselves that can be utilized to estimate the best possible path (ex. length of segment, uniqueness of models, etc.). Finding an appropriate estimation method using additional known attributes of the application would make this model-based in-sequence localization approach competitive with state-of-the-art stand-alone localization approaches. Thus extending the work in this direction would yield the most immediate practical results for implementation.

7.2.4 New Application Areas

Lastly, the development of this thesis is centered on finding an approach to localize a passenger vehicle using terrain data recorded by the vehicle. However, as shown in Chapters 5 and 6, the algorithms developed in this dissertation can be readily exploited in fields such as financial data, prediction of energy production, and inference of environmental data. Even this subset of applications is limited when considering the mountains of data that are currently collected from all aspects of life. For this reason, there are innumerable applications to which this research can be applied. The targeted implementation of in-sequence localization in any particular field is a promising area of research that can open new doors of discovery.

Appendix

The One Dimensional Chance Constrained Formulation

The material presented in this appendix was published in the 2014 American Control Conference [24]. This material is integral to the development in chapter 5 and for this reason it has been attached here for completeness.

1 Developing the Chance Constrained Formulation

Assuming additive white Gaussian noise, $\eta_j \sim N(0, \sigma_\eta^2)$, this section derives the probabilistic constraints used in the dynamical model extraction. First, we note that the addition of Gaussian noise results in Gaussian modeling errors, $e_{m,k} \sim N(\mu_{m,k}, \sigma_m^2)$. The mean $\mu_{m,k}$ and variance σ_m^2 of these errors is quantified in equations (.1) and (.2).

$$\begin{aligned}\mu_{e_{m,k}} =& \mathbb{E}[(d_k + \eta_k) - c_{m,1}(d_{k-1} + \eta_{k-1}) + ... \\ & + c_{m,N}(d_{k-N} + \eta_{k-N}] \\ =& d_k - c_{m,1}d_{k-1} + ... + c_{m,N}d_{k-N}\end{aligned} \tag{.1}$$

$$\begin{aligned}\sigma_{e_m}^2 = & var[(d_k + \eta_k) - c_{m,1}(d_{k-1} + \eta_{k-1}) + ... \\ & + c_{m,N}(d_{k-N} + \eta_{k-N}] = ...\end{aligned} \tag{.2}$$

$$= \sigma_\eta^2 + c_{m,1}^2\sigma_\eta^2 + ... + c_{m,N}^2\sigma_\eta^2$$
$$= \sigma_\eta^2(1 + \sum_{n=1}^{N} c_{m,n}^2).$$

Given the mean and variance of the modeling errors, and the probability distribution overlap shown in Fig. 5.3, the probabilities of error are shown in equations (.3) and (.4) where the first equation in (.4) is the probability of the transition point error exceeding ε to the positive side, and the second equation represents the probability of exceeding ε to the negative side.

$$P_{e_{m+1}} = \mathbb{P}(-\varepsilon \leq e_{m+1,\tau_m} \leq \varepsilon). \tag{.3}$$

$$\begin{aligned} P_{e_m} &= \mathbb{P}(e_{m,\tau_m} > \varepsilon), \\ P_{e_m} &= \mathbb{P}(e_{m,\tau_m} < -\varepsilon), \end{aligned} \tag{.4}$$

Representing the probability of the new segment error in terms of the standard normal distribution results in,

$$\Phi_{e_{m+1}}(\frac{\varepsilon - \mu_{e_{m+1,\tau_m}}}{\sigma_{e_{m+1}}}) - \Phi_{e_{m+1}}(\frac{-\varepsilon - \mu_{e_{m+1,\tau_m}}}{\sigma_{e_{m+1}}}). \tag{.5}$$

Correspondingly, the probability of exceeding the ε modeling error bound to the positive side can be expressed as,

$$1-\Phi_{e_m}(\frac{\varepsilon - \mu_{e_{m,\tau_m}}}{\sigma_{e_m}}), \tag{.6}$$

and to the negative side as,

$$\Phi_{e_m}(\frac{-\varepsilon - \mu_{e_{m,\tau_m}}}{\sigma_{e_m}}). \tag{.7}$$

Finally, these probabilities need to be expressed as constraints in optimization for the dynamical model extraction procedure in this paper. Using the result by

Kataoka [192] we maximize the probability that $|e_{m+1,k}| \leq \varepsilon$.

$$\sigma_{m+1}\Phi^{-1}_{e_{m+1}}(\frac{\rho}{2}+\frac{1}{2})+\mu_{e_{m+1,k}} \leq \varepsilon. \tag{.8}$$

Similarly, we express the transition point modeling error probabilities for the ending segment as,

$$\begin{aligned} \varepsilon &\leq \sigma_{e_m}\Phi^{-1}_{e_m}(1-\rho)+\mu_{e_{m,k}}, \\ -\varepsilon &\geq \sigma_{e_m}\Phi^{-1}_{e_m}(\rho)+\mu_{e_{m,k}}. \end{aligned} \tag{.9}$$

To maintain convexity in the problem, the variable p must be greater than 1/2. Working with the vehicle data, it was determined that a value of .99 or a 1% failure rate provided sufficient variety in the transition points. For this reason all figures shown represent results for data structures extracted to a $\rho = .99$ probability of correct detection.

Bibliography

[1] Dear, P. (1995) *Discipline and Experience: The Mathematical Way in the Scientific Revolution*, University of Chicago Press.

[2] Smolan, R. and J. Erwitt (2012) *The Human Face of Big Data*, Against All Odds Productions.

[3] Ahalt, S., D. Bedard, T. Carsey, J. Crabtree, K. Green, C. Jeffries, D. Knowles, H. Kum, H. Lander, N. Nassar, A. Rajasekar, and S. Thakur (2012) *Establishing a National Consortium for Data Science*, *Tech. rep.*, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA.

[4] Anguita, D., A. Ghio, L. Oneto, L. Parra, F. Xavier, R. Ortiz, and J. Luis (2012) "Human activity recognition on smartphones for mobile context awareness," in *Proceedings of the Neural Information Processing Conference*, Lake Tahoe, NV, pp. 1–9.

[5] Chen, B. and H. Cheng (2010) "A Review of the Applications of Agent Technology in Traffic and Transportation Systems," *Intelligent Transportation Systems, IEEE Transactions on*, **11**(2), pp. 485–497.

[6] Skog, I. and P. Handel (2009) "In-Car Positioning and Navigation Technologies - A Survey," *IEEE Transactions on Intelligent Transportation Systems*, **10**(1), pp. 4–21.

[7] Farrell, J. and M. Barth (1999) *The Global Positioning System & Inertial Navigation*, McGraw-Hill Companies.

[8] Drawil, N., H. Amar, and O. Basir (2013) "GPS Localization Accuracy Classification: A Context-Based Approach," *IEEE Transactions on Intelligent Transportation Systems*, **14**(1), pp. 262–273.

[9] Ochieng, W. Y., K. Sauer, D. Walsh, G. Brodin, S. Griffin, and M. Denney (2003) "GPS Integrity and Potential Impact on Aviation Safety," *The Journal of Navigation*, **56**, pp. 51–65.

[10] Dean, A. (2008) *Terrain-based Road Vehicle Localization using Attitude Measurements*, Ph.D. thesis, Department of Mechanical and Nuclear Engineering, Pennsylvania State University.

[11] Gustafsson, F., F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P. Nordlund (2002) "Particle Filters for Positioning, Navigation, and Tracking," *IEEE Transactions on Signal Processing*, **50**(2), pp. 425–437.

[12] Vemulapalli, P. K., V. Monga, and S. N. Brennan (2013) "Robust Extrema Features for Time-Series Data Analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **35**(6), pp. 1464–1479.

[13] Keogh, E. and S. Kasetty (2003) "On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration," *Data Mining and Knowledge Discovery*, **7**(4), pp. 349–371.

[14] Ding, H., G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh (2008) "Querying and mining of time series data: experimental comparison of representations and distance measures," *Proceedings of the VLDB Endowment*, **1**(2), pp. 1542–1552.

[15] Fu, T. (2011) "A review on time series data mining," *Engineering Applications of Artificial Intelligence*, **24**(1), pp. 164–181.

[16] Stoyanov, T., M. Magnusson, and A. J. Lilienthal (2013) "Comparative Evaluation of the Consistency of Three-dimensional Spatial Representations used in Autonomous Robot Navigation," *Journal of Field Robotics*, **30**(2), pp. 216–236.

[17] Mullane, J., E. Jose, M. D. Adams, and W. S. Wijesoma (2007) "Including probabilistic target detection attributes into map representations," *Robotics and Autonomous Systems*, **55**(1), pp. 72 – 85.

[18] Guerrero, J. L., J. Garcia, and J. M. Molina (2011) "Piecewise Linear Representation Segmentation in Noisy Domains with a Large Number of Measurements: The Air Traffic Control Domain," *International Journal on Artificial Intelligence Tools*, **20**(02), pp. 367–399.

[19] Skauli, T. (2011) "Sensor noise informed representation of hyperspectral data, with benefits for image storage and processing," *Opticas Express*, **19**(14), pp. 13031–13046.

[20] Laftchiev, E., C. Lagoa, and S. Brennan (2012) "Terrain-Based Vehicle Localization from Real-Time Data Using Dynamical Models," in *IEEE Conference on Decision and Control*, Maui, HI, pp. 3366–3371.

[21] ——— (2012) "Terrain-Based Vehicle Localization from Real-Time Data Using Dynamical Models," in *The Pennsylvania State University College of Engineering Research Symposium*, State College, PA, pp. 1–7.

[22] ——— (2014) "Vehicle Localization using in-Vehicle Pitch Data and Dynamical Models," *IEEE Transactions on Intelligent Transportation Systems*, **PP**(99), pp. 1–15, in-Press.

[23] ——— (2013) "Robust Map Design by Outlier Point Selection for Terrain-Based Vehicle Localization," in *IEEE Conference on Decision and Control*, Florence, Italy, pp. 2822–2827.

[24] ——— (2014) "Robust Data Map Design Using Chance Constrained Optimization," in *American Control Conference*, Portland, OR, pp. 4573–4580.

[25] ——— (2013) "Robust Map Design for Terrain-Based Vehicle Localization," in *The Pennsylvania State University College of Engineering Research Symposium*, State College, PA, pp. 1–6.

[26] ——— (2014) "Multi-Attribute Data Dynamics Discontinuity Identification: An Over-Bounding approach using One-Dimensional Probabilistic Constraints," in *The Pennsylvania State University College of Engineering Research Symposium*, State College, PA, pp. 1–7.

[27] Shatkay, H. and S. Zdonik (1996) "Approximate queries and representations for large data sequences," in *International Conference on Data Engineering*, New Orleans, LA, pp. 536–545.

[28] Keogh, E., K. Chakrabarti, M. Pazzani, and S. Mehrotra (2000) "Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases," *Journal of Knowledge and Information Systems*, **3**, pp. 263–286.

[29] Yi, B. and C. Faloutsos (2000) "Fast Time Sequence Indexing for Arbitrary Lp Norms," in *International Conference on Very Large Data Bases*, Cairo, Egypt, pp. 385–394.

[30] Chakrabarti, K., E. Keogh, S. Mehrotra, and M. Pazzani (2002) "Locally adaptive dimensionality reduction for indexing large time series databases," *ACM Transactions on Database Systems*, **27**(2), pp. 188–228.

[31] Keog, E. (1997) "Fast similarity search in the presence of longitudinal scaling in time series databases," in *International Conference on Tools with Artificial Intelligence*, Washington DC, USA, pp. 578–584.

[32] Smyth, P. and E. Keogh (1997) "Clustering and Mode Classification of Engineering Time Series Data," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 24–30.

[33] Oliver, J. and C. Forbes (1997) *Bayesian Approaches to Segmenting A Simple Time Series*, *Monash Econometrics and Business Statistics Working Papers 14*, Monash University, Department of Econometrics and Business Statistics.

[34] Guralnik, V. and J. Srivastava (1999) "Event detection from time series data," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Diego, California, USA, pp. 33–42.

[35] Srivastava, A. and A. Wieigend (1996) "Improved time series segmentation using gated experts with simulated annealing," in *IEEE International Conference on Neural Networks*, vol. 4, Washington, DC, pp. 1883–1888.

[36] Duncan, S. and G. Bryant (1996) "A new algorithm for segmenting data from time series," in *Proceedings of the IEEE Conference on Decision and Control*, vol. 3, Kobe, Japan, pp. 3123–3128.

[37] Papadopoulos, V. (1992) *Detection and Estimation of Abrupt Changes in Singals and Systems: A Dynamic Programming Approach*, Ph.D. thesis, University of London.

[38] Gustafsson, F. (1992) *Estimation of Discrete Parameters in Linear Systems*, Ph.D. thesis, Linkoping Studies in Science and Technology.

[39] Chu, C. (1995) "Time series segmentation: A sliding window approach," *Information Sciences*, **85**(13), pp. 147–173.

[40] Fancourt, C. and J. Principe (1996) "A Neighborhood Map of Competing One Step Predictors for Piecewise Segmentation and Identification of Time Series," in *Proceedings of the International Conference on Neural Networks*, vol. 4, Washington, DC, pp. 1906–1911.

[41] Keogh, E., S. Chu, D. Hart, and M. Pazzani (2001) "An online algorithm for segmenting time series," in *Proceedings IEEE International Conference on Data Mining*, San Jose, CA, pp. 289–296.

[42] Pavlidis, T. and S. Horowitz (1974) "Segmentation of Plane Curves," *IEEE Transactions on Computers*, **C-23**(8), pp. 860–870.

[43] Jia, P., H. He, and T. Sun (2008) "Error Restricted Piecewise Linear Representation of Time Series based on Special Points," in *World Congress on Intelligent Control and Automation*, Chongqing, China, pp. 2059–2064.

[44] Pratt, K. B. and E. Fink (2002) "Search for Patterns in Compressed Time Series," *International Journal of Image and Graphics*, **02**(01), pp. 89–106.

[45] Fink, E., K. Pratt, and H. Gandhi (2003) "Indexing of time series by major minima and maxima," in *Conference on Systems, Man and Cybernetics*, vol. 3, Washington, DC, pp. 2332–2335.

[46] Perng, C.-S., H. Wang, S. Zhang, and D. Parker (2000) "Landmarks: a new model for similarity-based pattern querying in time series databases," in *International Conference on Data Engineering*, San Diego, CA, pp. 33–42.

[47] Agrawal, R., C. Faloutsos, and A. Swami (1993) "Efficient Similarity Search In Sequence Databases," in *Proceedings of the International Conference on Foundations of Data Organization and Algorithms*, Chicago, IL, pp. 69–84.

[48] Faloutsos, C., M. Ranganathan, and Y. Manolopoulos (1993) *Fast subsequence matching in time-series databases*, *Tech. rep.*, University of Maryland at College Park, College Park, MD, USA.

[49] Moon, Y., K. Whang, and W. Loh (2001) "Duality-Based Subsequence Matching in Time-Series Databases," in *Proceedings of the International Conference on Data Engineering*, Heidelberg, Germany, pp. 263–272.

[50] Kim, S. and B. Jeong (2007) "Performance bottleneck of subsequence matching in time-series databases: Observation, solution, and performance evaluation," *Information Sciences*, **177**(22), pp. 4841–4858.

[51] Kim, S., S. Park, and W. Chu (2001) "An index-based approach for similarity search supporting time warping in large sequence databases," in *Proceedings of the International Conference on Data Engineering*, Heidelberg, Germany, pp. 607–614.

[52] Lim, S., H. Park, and S. Kim (2007) "Using multiple indexes for efficient subsequence matching in time-series databases," *Information Sciences*, **177**(24), pp. 5691–5706.

[53] Morinaka, Y., T. Amagasa, and S. Uemura (2001) "The L-index: An indexing structure for efficient subsequence matching in time sequence databases," in *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Hong Kong, China, pp. 51–60.

[54] Keogh, E., J. Lin, and A. Fu (2005) "HOT SAX: efficiently finding the most unusual time series subsequence," in *IEEE International Conference on Data Mining*, Houston, TX, pp. 226–233.

[55] Wu, H., B. Salzberg, and D. Zhang (2004) "Online event-driven subsequence matching over financial data streams," in *Proceedings of the ACM SIGMOD international conference on Management of data*, SIGMOD, Paris, France, pp. 23–34.

[56] Wu, H., B. Salzberg, G. Sharp, S. Jiang, H. Shirato, and D. Kaeli (2005) "Subsequence matching on structured time series data," in *Proceedings of the ACM SIGMOD international conference on Management of data*, SIGMOD, Baltimore, Maryland, pp. 682–693.

[57] Gusfield, D. (1997) *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*, Cambridge University Press.

[58] Park, S., D. Lee, and W. Chu (1999) "Fast Retrieval of Similar Subsequences in Long Sequence Databases," in *Proceedings of the Workshop on Knowledge and Data Engineering Exchange*, KDEX, Chicago, IL, pp. 60–68.

[59] Kim, S., J. Yoon, S. Park, and T. Kim (2002) "Shape-based retrieval of similar subsequences in time-series databases," in *Proceedings of the ACM symposium on Applied computing*, SAC, Madrid, Spain, pp. 438–445.

[60] Fu, T., F. Chung, V. Ng, and R. Luk (2001) "Pattern Discovery from Stock Time Series Using Self-Organizing Maps," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Disvoery and Data Mining Workshop on Temporal Data Mining*, San Francisco, CA, pp. 27–37.

[61] Kohonen, T. (2001) *Self-Organizing Maps*, Physics and astronomy online library, Springer Berlin Heidelberg.

[62] ——— (1982) "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, **43**(1), pp. 59–69.

[63] Ripley, B. (2007) *Pattern Recognition and Neural Networks*, Cambridge University Press.

[64] Euliano, N. and J. Principe (1996) "Spatio-temporal self-organizing feature maps," in *IEEE International Conference on Neural Networks*, vol. 4, Washington,DC, pp. 1900–1905.

[65] Mörchen, F. and A. Ultsch (2005) "Optimizing time series discretization for knowledge discovery," in *Proceedings of the ACM SIGKDD international conference on Knowledge discovery in data mining*, KDD, Chicago, IL, pp. 660–665.

[66] Ultsch, A. (1999) *Data Mining and Knowledge Discovery with Emergent Self-Organizing Feature Maps for Multivariate Time Series*, Elsevier.

[67] Wang, X., K. Smith, and R. Hyndman (2005) "Dimension Reduction for Clustering Time Series Using Global Characteristics," in *Computational Science ICCS 2005*, vol. 3516 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 792–795.

[68] Kuo, S., S. Li, Y. Cheng, and M. Ho (2004) "Knowledge discovery with SOM networks in financial investment strategy," in *International Conference on Hybrid Intelligent Systems*, pp. 98–103.

[69] Guo, X., X. Liang, and N. Li (2007) "Automatically Recognizing stock patterns using RPCL neural Networks," in *Proceedings of the International Conference on Intelligent Systems and Knowledge Engineering*, pp. 997–1004.

[70] Kalpakis, K., D. Gada, and V. Puttagunta (2001) "Distance measures for effective clustering of ARIMA time-series," in *Proceedings IEEE International Conference on Data Mining*, San Jose, CA, pp. 273–280.

[71] Bagnall, A. and G. Janacek (2005) "Clustering Time Series with Clipped Data," *Machine Learning*, **58**(2–3), pp. 151–178.

[72] Xiong, Y. and D. Yeung (2004) "Time series clustering with {ARMA} mixtures," *Pattern Recognition*, **37**(8), pp. 1675 – 1689.

[73] Bettini, C., X. Wang, S. Jajodia, and J. Lin (1998) "Discovering frequent event patterns with multiple granularities in time sequences," *IEEE Transactions on Knowledge and Data Engineering*, **10**(2), pp. 222–237.

[74] Li, Y., X. Wang, and S. Jajodia (2001) "Discovering Temporal Patterns in Multiple Granularities," in *Proceedings of the First International Workshop on Temporal, Spatial, and Spatio-Temporal Data Mining-Revised Papers*, TSDM '00, pp. 5–19.

[75] Geurts, P. (2001) "Pattern Extraction for Time Series Classification," in *Principles of Data Mining and Knowledge Discovery*, vol. 2168 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 115–127.

[76] Zhang, H., T. Ho, and M. Lin (2004) "A Non-parametric Wavelet Feature Extractor for Time Series Classification," in *Advances in Knowledge Discovery and Data Mining*, vol. 3056 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 595–603.

[77] Povinelli, R., M. Johnson, A. Lindgren, and J. Ye (2004) "Time series classification using Gaussian mixture models of reconstructed phase spaces," *IEEE Transactions on Knowledge and Data Engineering*, **16**(6), pp. 779–783.

[78] Rodríguez, J. J. and C. J. Alonso (2004) "Interval and dynamic time warping-based decision trees," in *Proceedings of the ACM symposium on Applied computing*, SAC, Nicosia, Cyprus, pp. 548–552.

[79] Xi, X., E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana (2006) "Fast time series classification using numerosity reduction," in *Proceedings of the 23rd international conference on Machine learning*, ICML, Pittsburgh, Pennsylvania, pp. 1033–1040.

[80] Wei, L. and E. Keogh (2006) "Semi-supervised time series classification," in *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD, Philadelphia, PA, USA, pp. 748–753.

[81] Povinelli, R. and X. Feng (1999) "Data Mining Of Multiple Nonstationary Time Series," in *Proceedings of Artifical Neural Netowrks in Engineering*, pp. 511–516.

[82] Kahveci, T., A. Singh, and A. Gurel (2002) "Similarity searching for multi-attribute sequences," in *Conference on Scientific and Statistical Database Management*, Edinburgh, Scotland, pp. 175–184.

[83] Lee, A. J., H.-W. Wu, T.-Y. Lee, Y.-H. Liu, and K.-T. Chen (2009) "Mining closed patterns in multi-sequence time-series databases," *Data & Knowledge Engineering*, **68**(10), pp. 1071–1090.

[84] Minnen, D., C. Isbell, I. Essa, and T. Starner (2007) "Detecting Subdimensional Motifs: An Efficient Algorithm for Generalized Multivariate Pattern Discovery," in *IEEE International Conference on Data Mining*, Omaha, Nebraska, pp. 601–606.

[85] Minnen, D., C. L. Isbell, I. Essa, and T. Starner (2007) "Discovering Multivariate Motifs Using Subsequence Density Estimation and Greedy Mixture Learning," in *Conference on Artificial Intelligence*, Vancouver, British Columbia, Canada, pp. 615–620.

[86] Plant, C., A. Wohlschlager, and A. Zherdin (2009) "Interaction-Based Clustering of Multivariate Time Series," in *IEEE International Conference on Data Mining*, Miami, Florida, pp. 914–919.

[87] Tatavarty, G., R. Bhatnagar, and B. Young (2007) "Discovery of Temporal Dependencies between Frequent Patterns in Multivariate Time Series," in *IEEE Symposium on Computational Intelligence and Data Mining*, Honolulu, Hawaii, pp. 688–696.

[88] Wang, X., L. Wang, and A. Wirth (2008) "Pattern discovery in motion time series via structure-based spectral clustering," in *IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, AK, pp. 1–8.

[89] Shibuya, T., T. Harada, and Y. Kuniyoshi (2009) "Causality quantification and its applications: structuring and modeling of multivariate time series," in *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD, Paris, France, pp. 787–796.

[90] Matsubara, Y., Y. Sakurai, and M. Yoshikawa (2009) "Scalable Algorithms for Distribution Search," in *International Conference on Data Mining*, Miami, FL, pp. 347–356.

[91] Moreira, J., C. Ribeiro, J.-M. Saglio, and M. Scholl (2008) "A model of approximations for representing time-varying multidimensional data," in *International Conference on Data Engineering Workshop*, Cancun, Mexico, pp. 113–120.

[92] Gonzalez, H., J. Han, Y. Ouyang, and S. Seith (2011) "Multidimensional Data Mining of Traffic Anomalies on Large-Scale Road Networks," in *Transportation Research Record*, pp. 75–84.

[93] Randen, T. and J. Husøy (1999) "Filtering for texture classification: a comparative study," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **21**(4), pp. 291–310.

[94] Zimmermann, K., J. Matas, and T. Svoboda (2009) "Tracking by an Optimal Sequence of Linear Predictors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **31**(4), pp. 677–692.

[95] Al-Takrouri, S. and A. V. Savkin (2010) "A model validation approach to texture recognition and inpainting," *Pattern Recognition*, **43**(6), pp. 2054–2067.

[96] Ashraf, A. B., S. Lucey, and T. Chen (2010) "Reinterpreting the Application of Gabor Filters as a Manipulation of the Margin in Linear Support Vector Machines," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **32**(7), pp. 1335–1341.

[97] Turaga, P., R. Chellappa, and A. Veeraraghavan (2010) "Advances in Video-Based Human Activity Analysis: Challenges and Approaches," in *Advances in Computers*, vol. 80 of *Advances in Computers*, Elsevier, pp. 237–290.

[98] Kandaswamy, U., S. Schuckers, and D. Adjeroh (2011) "Comparison of Texture Analysis Schemes Under Nonideal Conditions," *IEEE Transactions on Signal Processing*, **20**(8), pp. 2260–2275.

[99] Sekita, I., T. Kurita, and N. Otsu (1992) "Complex autoregressive model for shape recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **14**(4), pp. 489–496.

[100] Potocnik, P., E. Govekar, and I. Grabec (2006) "Application of psychoacoustic filtering for machine fault detection," *International Journal of Materials and Product Technology*, **27**(3–4), pp. 229–237.

[101] Graciarena, M. and H. Franco (2003) "Unsupervised noise model estimation for model-based robust speech recognition," in *IEEE Workshop on Automatic Speech Recognition and Understanding*, St Thomas, VI, pp. 351–356.

[102] Shenoy, A., Y. Wu, and Y. Wang (2005) "Singing voice detection for karaoke application," in *Visual Communications and Image Processing*, vol. 5960, 1-4, Beijing, China, pp. 752–762.

[103] Johnsen, M. and A. Canterla (2012) "Joint feature and model training for minimum detection errors applied to speech subword detection," in *IEEE International Workshop on Machine Learning for Signal Processing*, Santander, Spain, pp. 1–6.

[104] Jain, A., S. Prabhakar, L. Hong, and S. Pankanti (2000) "Filterbank-based fingerprint matching," *IEEE Transactions on Image Processing*, **9**(5), pp. 846–859.

[105] Bissacco, A., A. Chiuso, Y. Ma, and S. Soatto (2001) "Recognition of human gaits," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, Kauai, HI, pp. 52–57.

[106] Veeraraghavan, A., A. Roy-Chowdhury, and R. Chellappa (2005) "Matching shape sequences in video with applications in human movement analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **27**(12), pp. 1896–1909.

[107] Mazzaro, C., M. Sznaier, O. Camps, S. Soatto, and A. Bissacco (2002) "A model (In)validation approach to gait recognition," in *International Symposium on 3D Data Processing Visualization and Transmission*, Padua, Italy, pp. 700 – 703.

[108] Saisan, P., G. Doretto, Y. Wu, and S. Soatto (2001) "Dynamic texture recognition," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Kauai, HI, pp. 58–63.

[109] Bissacco, A., A. Chiuso, and S. Soatto (2007) "Classification and Recognition of Dynamical Models: The Role of Phase, Independent Components, Kernels and Optimal Transport," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **29**(11), pp. 1958–1972.

[110] Ravichandran, A., R. Chaudhry, and R. Vidal (2013) "Categorizing Dynamic Textures Using a Bag of Dynamical Systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **35**(2), pp. 342–353.

[111] Li, R., T.-P. Tian, and S. Sclaroff (2012) "Divide, Conquer and Coordinate: Globally Coordinated Switching Linear Dynamical System," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **34**(4), pp. 654–669.

[112] Martin, R. (2000) "A metric for ARMA processes," *IEEE Transactions on Signal Processing*, **48**(4), pp. 1164–1170.

[113] Borenstein, J., H. R. Everett, L. Feng, and D. Wehe (1997) "Mobile Robot Positioning: Sensors and Techniques," *Journal of Robotic Systems*, **14**(4), pp. 231–249.

[114] Abbott, H. and D. Powell (1999) "Land-vehicle navigation using GPS," *Proceedings of the IEEE*, **87**(1), pp. 145–162.

[115] Borenstein, J. and F. Liqiang (1996) "Measurement and Correction of Systematic Odometry Errors in Mobile Robots," *IEEE Transactions on Robotics and Automation*, **12**(6), pp. 869–880.

[116] Doh, N., H. Choset, and W. Chung (2006) "Relative Localization using Path Odometry Information," *Autonomous Robots*, **21**(2), pp. 143–154.

[117] Carlson, C., J. Gerdes, and J. Powell (2002) "Practical Position and Yaw Rate Estimation with GPS and Differential Wheelspeeds," in *Proceedings of the International Symposium on AVEC*, Hiroshima, Japan, pp. 1–8.

[118] ——— (2004) "Error Sources When Land Vehicle Dead Reckoning with Differential Wheelspeeds," *Journal of The Institute of Navigation*, **51**(1), pp. 13–28.

[119] Martini, R. (2006) *GPS/INS Sensing Coordination for Vehicle State Identification and Road Grade Positioning*, Master's thesis, Department of Mechanical and Nuclear Engineering, Pennsylvania State University.

[120] Gupta, V. (2009) *Vehicle Localization Using Low-accuracy GPS, IMU and Map-aided Vision*, Ph.D. thesis, Department of Mechanical and Nuclear Engineering, Pennsylvania State University.

[121] Borenstein, J. and L. Feng (1996) "Gyrodometry: A new method for combining data from gyros and odometry in mobile robots," in *IEEE International Conference on Robotics and Automation*, vol. 1, Minneapolis, MN, pp. 423–428.

[122] Bernstein, D. and A. Kornhauser (1996) *An Introduction to Map Matching for Personal Navigation Assistants*, *Tech. rep.*, Transportation Research Board.

[123] Quddus, M. A., W. Y. Ochieng, and R. B. Noland (2006) "Integrity of map-matching algorithms," *Transportation Research Part C: Emerging Technologies*, **14**(4), pp. 283 – 302.

[124] ——— (2007) "Current map-matching algorithms for transport applications: State-of-the art and future research directions," *Transportation Research Part C: Emerging Technologies*, **15**(5), pp. 312–328.

[125] Smith, R., M. Self, and P. Cheeseman (1988) "A stochastic map for uncertain spatial relationships," in *The fourth international symposium on Robotics Research*, MIT Press, Cambridge, MA, pp. 467–474.

[126] Quddus, M., W. Ochieng, L. Zhao, and R. Noland (2003) "A general map matching algorithm for transport telematics applications," *GPS Solutions*, **7**(3), pp. 157–167.

[127] Dean, A., R. Martini, and S. Brennan (2008) "Terrain-Based Road Vehicle Localization Using Particle Filters," in *Proceedings of the American Control Conference*, Seattle, WA, pp. 236–241.

[128] Leonard, J. and H. Durrant-Whyte (1991) "Mobile Robot Localization by Tracking Geometric Beacons," *IEEE Transactions on Robotics and Automation*, **7**(3), pp. 376–382.

[129] Bosse, M., D. Koller, Z. Ghahramani, H. Durrant-Whyte, and A. Ng (2009) "Keypoint design and evaluation for place recognition in 2D lidar maps," *Journal of Robotics and Autonomous Systems*, **57**(12), pp. 1211–1224.

[130] Mikolajczyk, K. and H. Uemura (2008) "Action recognition with motion-appearance vocabulary forest," in *IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, AK, pp. 1–8.

[131] Thrun, S. and R. Zlot (2004) "Reduced sift features for image retrieval and indoor localization," in *Australian Conference on Robotics and Automation*, Canberra, Australia, pp. 1–8.

[132] Smith, R. and P. Cheeseman (1986) "On the representation and estimation of spatial uncertainty," *International Journal of Robotics Research*, **5**(4), pp. 56–68.

[133] Smith, R., M. Self, and P. Cheeseman (1990) *Estimating uncertain spatial relationships in robotics*, chap. Autonomous Robot Vehicles, Springer.

[134] Moutarlier, P. and R. Chatila (1990) "An experimental system for incremental environment modeling by an autonomous mobile robot," in *Proceedings of The First International Symposium on Experimental Robotics*, London, UK, pp. 327–346.

[135] Levinson, J. and S. Thrun (2010) "Robust Vehicle Localization in Urban Environments Using Probabilistic Maps," in *IEEE International Conference on Robotics and Automation*, Anchorage, AK, pp. 4372–4378.

[136] Törnqvist, D., T. Schn, R. Karlsson, and F. Gustafsson (2009) "Particle Filter SLAM with High Dimensional Vehicle Model," *Journal of Intelligent & Robotic Systems*, **55**(4-5), pp. 249–266.

[137] Vemulapalli, P., A. Dean, and S. Brennan (2011) "Pitch-based Vehicle Localization using Time Series Subsequence Matching with Multi-scale Extrema Features," in *American Control Conference*, San Francisco, CA, pp. 2405–2410.

[138] Kadetodat, S., P. Vemulapalli, S. Brennan, and C. Lagoa (2011) "Terrain-Aided Localization using Feature-Based Particle Filtering," in *Dynamic Systems and Control Conference*, Arlington, VA, pp. 725–731.

[139] Dissanayke, M., P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba (2001) "A Solution to the Simultaneous Localization and Map Building (SLAM) Problem," *IEEE Transactions on Robotics and Control*, **17**(3), pp. 229–241.

[140] Hasberg, C., S. Hensel, and C. Stiller (2012) "Simultaneous Localization and Mapping for Path-Constrained Motion," *IEEE Transactions on Intelligent Transportation Systems*, **13**(2), pp. 541–552.

[141] Brown, A. and Y. Lu (2004) "Performance Test Results of an Integrated GPS/MEMS Inertial Navigation Package," in *Proceedings of the International Technical Meeting of the Satellite Division*, vol. 17, Long Beach, CA, pp. 825–832.

[142] Godha, S. and M. Cannon (2007) "GPS/MEMS INS integrated system for navigation in urban areas," *GPS Solutions*, **11**(3), pp. 193–203.

[143] El-Sheimy, N., H. Hou, and X. Niu (2008) "Analysis and Modeling of Inertial Sensors Using Allan Variance," *IEEE Transactions on Instrumentation and Measurement*, **57**(1), pp. 140–149.

[144] El-Sheimy, N., S. Nassar, and A. Noureldin (2004) "Wavelet denoising for IMU alignment," *IEEE Aerospace and Electronic Systems Magazine*, **19**(10), pp. 32–39.

[145] Aydemir, G. and A. Saranli (2012) "Characterization and calibration of MEMS inertial sensors for state and parameter estimation applications," *Measurement*, **45**(5), pp. 1210–1225.

[146] Jerath, K. and S. Brennan (2011) "GPS-Free Terrain-based Vehicle Tracking Performance as a Function of Inertial Sensor Characteristics," in *Dynamics Systems and Control Conference*, Arlington, VA, pp. 367–374.

[147] Aggarwal, P., Z. Syed, X. Niu, and N. El-Sheimy (2007) "Thermal Calibration of Low Cost MEMS Sensors for Integrated Positioning Navigation Systems," in *Proceedings of the National Technical Meeting of The Institute of Navigation*, vol. 1, San Diego, CA, pp. 343–349.

[148] Skog, I. and P. Händel (2006) "Calibration of a MEMS inertial measurement unit," in *Proceedings of the IMEKO World Congress*, vol. 17, Rio de Janeiro, Brazil, pp. 1–6.

[149] Wang, L., Y. Hao, and F. Wang (2011) "Calibration of low cost MEMS inertial Measurement Unit for an FPGA-based navigation system," in *IEEE International Conference on Information and Automation*, Shenzhen, China, pp. 181–186.

[150] Isermann, R. and P. Ball (1997) "Trends in the application of model-based fault detection and diagnosis of technical processes," *Control Engineering Practice*, **5**(5), pp. 709–719.

[151] Venkatasubramanian, V., R. Rengaswamy, K. Yin, and S. Kavuri (2003) "A review of process fault detection and diagnosis: Part I: Quantitative model-based methods," *Computers & Chemical Engineering*, **27**(3), pp. 293–311.

[152] Venkatasubramanian, V., R. Rengaswamy, and S. Kavuri (2003) "A review of process fault detection and diagnosis: Part II: Qualitative models and search strategies," *Computers & Chemical Engineering*, **27**(3), pp. 313 – 326.

[153] Venkatasubramanian, V., R. Rengaswamy, S. Kavuri, and K. Yin (2003) "A review of process fault detection and diagnosis: Part III: Process history based methods," *Computers & Chemical Engineering*, **27**(3), pp. 327–346.

[154] Hwang, I., S. Kim, Y. Kim, and C. Seah (2010) "A Survey of Fault Detection, Isolation, and Reconfiguration Methods," *IEEE Transactions on Control Systems Technology*, **18**(3), pp. 636–653.

[155] Isermann, R. (2006) "Fault detection with parity equations," in *Fault-Diagnosis Systems*, Springer Berlin Heidelberg, pp. 197–229.

[156] Thrun, S., W. Burgard, and D. Fox (2005) *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*, The MIT Press.

[157] Niedfeldt, P., D. Kingston, and R. Beard (2011) "Vehicle state estimation within a road network using a Bayesian filter," in *Proceedings of the American Control Conference*, San Fransisco, CA, pp. 4910–4915.

[158] Worrall, S. and E. Nebot (2007) "Using Non-Parametric Filters and Sparse Observations to Localise a Fleet of Mining Vehicles," in *IEEE International Conference on Robotics and Automation*, Rome, Italy, pp. 509–516.

[159] Tang, Z. and U. Ozguner (2003) "Sensor fusion for target track maintenance with multiple UAVs based on Bayesian filtering method and Hospitability Map," in *IEEE Conference on Decision and Control*, Maui, HI, pp. 19–24.

[160] Jemmott, C., R. Culver, and N. Bose (2008) "Passive sonar target localization using a histogram filter with model-derived priors," in *Conference on Signals, Systems and Computers*, Pacific Grove, CA, pp. 283–287.

[161] Chan, K.-P. and A.-C. Fu (1999) "Efficient time series matching by wavelets," in *Proceedings of the International Conference onData Engineering*, pp. 126–133.

[162] Cai, Y. and R. Ng (2004) "Indexing Spatio-temporal Trajectories with Chebyshev Polynomials," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Paris, France, pp. 599–610.

[163] Chakrabarti, K., E. Keogh, S. Mehrotra, and M. Pazzani (2002) "Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases," *ACM Transactions on Database Systems*, **27**(2), pp. 188–228.

[164] Kung, C.-P. and C.-S. Lin (1996) "Parallel sequence fault simulation for synchronous sequential circuits," *Journal of Electronic Testing*, **9**(3), pp. 267–277.

[165] Tsai, C.-F. and Y.-C. Hsiao (2010) "Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches," *Decision Support Systems*, **50**(1), pp. 258 – 269.

[166] Ni, J., C. V. Ravishankar, and S. Member (2007) "Indexing Spatiotemporal Trajectories with Efficient Polynomial Approximation," *IEEE Transactions on Knowledge and Data Engineering*, **19**(5), pp. 1–16.

[167] Laftchiev, E., C. Lagoa, and S. Brennan (2014) "Noise Resistant AR Model-Based Data Structures for In-Sequence Localization," *IEEE Transactions on Intelligent Transportation Systems*, #(#), p. #, in Review.

[168] Källhammer, J.-E., H. Pettersson, D. Eriksson, S. Junique, S. Savage, C. Vieider, J. Y. Andersson, J. Franks, J. Van Nylen, H. Vercammen, T. Kvisterø y, F. Niklaus, and G. Stemme (2006) "Fulfilling the pedestrian protection directive using a long-wavelength infrared camera designed to meet both performance and cost targets," in *Proceedings of SPIE*, vol. 6198, 09, pp. 619809–1 – 619809–11.

[169] Fang, H., M. Yang, R. Yang, and C. Wang (2009) "Ground-Texture-Based Localization for Intelligent Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, **10**(3), pp. 463–468.

[170] Parra Alonso, I., D. Fernandez Llorca, M. Gavilan, S. Alvarez Pardo, M. Garcia-Garrido, L. Vlacic, and M. Sotelo (2012) "Accurate Global Localization Using Visual Odometry and Digital Maps on Urban Environments," *IEEE Transactions on Intelligent Transportation Systems*, **13**(4), pp. 1535–1545.

[171] Sivaraman, S. and M. Trivedi (2013) "Integrated Lane and Vehicle Detection, Localization, and Tracking: A Synergistic Approach," *IEEE Transactions on Intelligent Transportation Systems*, **14**(2), pp. 906–917.

[172] Jo, K., K. Chu, and M. Sunwoo (2012) "Interacting Multiple Model Filter-Based Sensor Fusion of GPS With In-Vehicle Sensors for Real-Time Vehicle Positioning," *IEEE Transactions on Intelligent Transportation Systems*, **13**(1), pp. 329–343.

[173] Grewal, M., L. Weill, and A. Andrews (2007) *Global Positioning Systems, Inertial Navigation, and Integration*, Wiley.

[174] Wu, Z., M. Yao, H. Ma, and W. Jia (2013) "Improving Accuracy of the Vehicle Attitude Estimation for Low-Cost INS/GPS Integration Aided by the GPS-Measured Course Angle," *IEEE Transactions on Intelligent Transportation Systems*, **14**(2), pp. 553–564.

[175] Schindler, A. (2013) "Vehicle self-localization with high-precision digital maps," in *IEEE Intelligent Vehicles Symposium*, Gold Coast, QLD, pp. 141–146.

[176] Roth, J., T. Schaich, and G. Trommer (2012) "Cooperative GNSS-based method for vehicle positioning," *Gyroscopy and Navigation*, **3**(4), pp. 245–254.

[177] Schleicher, D., L. Bergasa, M. Ocana, R. Barea, and M. Lopez (2009) "Real-Time Hierarchical Outdoor SLAM Based on Stereovision and GPS Fusion," *IEEE Transactions on Intelligent Transportation Systems*, **10**(3), pp. 440–452.

[178] Ramanandan, A., A. Chen, and J. Farrell (2012) "Inertial Navigation Aiding by Stationary Updates," *IEEE Transactions on Intelligent Transportation Systems*, **13**(1), pp. 235–248.

[179] Bosch, P. and A. Van der Klauw (1994) *Modeling: identification and simulation of dynamical systems*, CRC Press.

[180] Ozay, N., M. Sznaier, C. Lagoa, and O. Camps (2012) "A Sparsification Approach to Set Membership Identification of Switched Affine Systems," *IEEE Transactions on Automatic Control*, **57**(3), pp. 634–648.

[181] Dissanayke, M., S. Williams, H. Durrant-Whyte, and T. Bailey (2002) "Map Management for Efficient Simultaneous Localization and Mapping (SLAM)," *Autonomous Robots*, **12**(3), pp. 267–286.

[182] Thrun, S., Y. Liu, D. Koller, A. Ng, Z. Ghahramani, and H. Durrant-Whyte (2004) "Simultaneous Mapping and Localization With Sparse Extended Information Filters," *International Journal of Robotics Research*, **23**(7-8), pp. 693–716.

[183] Guivant, J. and E. Nebot (2001) "Optimization of the Simultaneous Localization and Map-Building Algorithm for Real-Time Implementation," *IEEE Transactions on Robotics and Automation*, **17**(3), pp. 242–257.

[184] Board, I.-S. S. (1998) "IEEE Standard Specification Format Guide and Test Procedure for Single-Axis Interferometric Fiber Optic Gyros," *IEEE Std 952-1997*, pp. 1–83.

[185] Washi, T., A. Kawamura, and Y. Iiguni (2006) "Sinusoidal Noise Reduction Method Using Leaky LMS Algorithm," in *International Symposium on Intelligent Signal Processing and Communications*, Yonago, Japan, pp. 303–306.

[186] Hou, H. (2004) *Modeling Inertial Sensors Errors Using Allan Variance*, *Tech. rep.*, University of Calgary.

[187] Grigorie, T. L., R. Obreja, D. G. Sandu, and J. I. Corcau (2012) "Allan variance Analysis of the Miniaturized Sensors in a Strap-Down Inertial Measurement Unit," in *International Multidisciplinary Scientific Geo-Conference*, pp. 443–450.

[188] Ozay, N., M. Sznaier, C. Lagoa, and O. Camps (2008) "A Sparsification Approach to Set Membership Identification of a Class of Affine Hybrid Systems," in *IEEE Conference on Decision and Control*, Cancun, Mexico, pp. 123–130.

[189] Gama, J., R. Fernandes, and R. Rocha (2006) "Decision trees for mining data streams," *Intelligent Data Analysis*, **10**(1), pp. 23–45.

[190] Van Trees, H. (2004) *Detection, Estimation, and Modulation Theory*, 1, Wiley.

[191] Isermann, R. (2006) *Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance*, Springer.

[192] Kataoka, S. (1963) "A Stochastic Programming Model," *Econometrica*, **31**(1-2), pp. 181–196.

[193] Bouzgou, H. (2014) "A fast and accurate model for forecasting wind speed and solar radiation time series based on extreme learning machines and principal components analysis," *Journal of Renewable and Sustainable Energy*, **6**(1), pp. –.

[194] Giebel, G., R. Brownsword, G. Kariniotakis, M. Denhard, and C. Draxl (2011) *The State of the Art in Short-Term Prediction of Wind Power*, *Literature Review 2.0*, ANEMOS.plus.

[195] Neumann, D., B. Rajagopalan, and E. Zagona (2003) "egression Model for Daily Maximum Stream Temperature," *Journal of Environmental Engineering*, **129**(7), pp. 667–674.

[196] Kelleher, C., T. Wagener, M. Gooseff, B. McGlynn, K. McGuire, and L. Marshall (2012) "Investigating controls on the thermal sensitivity of Pennsylvania streams," *Hydrological Processes*, **26**(5), pp. 771–785.

[197] Null, S. E., M. L. Deas, and J. R. Lund (2010) "Flow and water temperature simulation for habitat restoration in the Shasta River, California," *River Research and Applications*, **26**(6), pp. 663–681.

[198] Nemirovski, A. and A. Shapiro (2006) "Convex Approximations of Chance Constrained Programs," *SIAM Journal on Optimization*, **17**(4), pp. 969–996.

[199] Ermoliev, Y. and R. J.-B. Wets (1988) *Numerical techniques for stochastic optimization*, Springer-Verlag.

[200] Jasour, A. and C. Lagoa (2012) "Semidefinite relaxations of chance constrained algebraic problems," in *IEEE Conference on Decision and Control*, Maui,HI, pp. 2527–2532.

[201] Jasour, A., N. Aybat, and C. Lagoa. (2014) "Semidefinite programming for chance optimization over semialgebraic sets," *arXiv*, (1402.6382).

[202] Lasserre, J. B. (2000) "Global Optimization with Polynomials and the Problem of Moments," *SIAM Journal on Optimization*, **11**(3), pp. 796–817.

[203] Lasserre, J. (2010) *Moments, Positive Polynomials and Their Applications*, no. 10 in Imperial College Press optimization series, vol. 1, Imperial College Press.

[204] Kall, P. and J. Mayer (2010) *Stochastic Linear Programming: Models, Theory, and Computation*, Springer.

[205] Lew, D., G. Brinkman, E. Ibanez, A. Florita, M. Heaney, B.-M. Hodge, M. Hummon, G. Stark, J. King, S. Lefton, N. Kumar, D. Agan, G. Jordan, and S. Venkataraman (2011) *The Western Wind and Solar Integration Study Phase 2, Literature Review 2.0*, NREL.

[206] Baïle, R., J. F. Muzy, and P. Poggi (2011) "Short-term forecasting of surface layer wind speed using a continuous random cascade model," *Wind Energy*, **14**(6), pp. 719–734.

[207] Brown, B., R. Katz, and A. Murphy (1984) "Time Series Models to Simulate and Forecast Wind Speed and Wind Power," *Journal of Climate and Applied Meteorology*, **23**(8), pp. 1184–1195.

[208] Hammer, A., D. Heinemann, E. Lorenz, and B. Lckehe (1999) "Short-term forecasting of solar radiation: a statistical approach using satellite data," *Solar Energy*, **67**(13), pp. 139 – 150.

[209] Daniel, A. and A. Chen (1991) "Stochastic simulation and forecasting of hourly average wind speed sequences in Jamaica," *Solar Energy*, **46**(1), pp. 1 – 11.

[210] Kay, J., R. Handcock, A. Gillespie, C. Konrad, S. Burges, N. Naveh, and D. Booth (2001) "Stream-temperature estimation from thermal infrared images," in *IEEE Geoscience and Remote Sensing Symposium*, vol. 1, Sydney, Australia, pp. 112–114.

[211] Jensen, A., B. Neilson, M. McKee, and Y. Chen (2012) "Thermal remote sensing with an autonomous unmanned aerial remote sensing platform for surface stream temperatures," in *IEEE Geoscience and Remote Sensing Symposium*, Munich, Germany, pp. 5049–5052.

[212] Handcock, R., A. Gillespie, K. Cherkauer, J. Kay, S. Burges, and S. Kampf (2006) "Accuracy and uncertainty of thermal-infrared remote sensing of stream temperatures at multiple spatial scales," *Remote Sensing of Environment*, **100**(4), pp. 427 – 440.

[213] Kwasniok, F. and L. A. Smith (2004) "Real-Time Construction of Optimized Predictors from Data Streams," *Phys. Rev. Lett.*, **92**, p. 164101.

[214] Chen, Y. (2008) *Towards a Unified Framework for Efficient Access Methods and Query Operations in Spatio-Temporal Databases*, Ph.D. thesis, University of Michigan.

Vita

Emil Laftchiev

Education:

- B.S. in Electrical Engineering, The Pennsylvania State University (Honors in Electrical Engineering), May 2007

Professional Experience:

- Graduate Lecturer, The Pennsylvania State University, Summer 2014-Fall 2014.
- Graduate Teaching Assistant, The Pennsylvania State University, Fall 2010-Spring 2014.
- Graduate Research Assistant, The Pennsylvania State University, Fall 2007-Spring 2010.
- SAP America Support Intern, Newtown Square, PA Summer 2004
- SAP America Hosting Intern, Newtown Square, PA Summer 2006

Publications:

- Laftchiev, E., Lagoa, C., Brennan, S. 2014. "Vehicle Localization using in-Vehicle Pitch Data and Dynamical Models" IEEE Transactions on Intelligent Transportation Systems, *In-Press*.
- Laftchiev, E., Lagoa, C., Brennan, S. 2014. "Robust Data Map Design Using Chance Constrained Optimization, Portland, OR, pp. 4573-4580.
- Laftchiev, E., Lagoa, C., Brennan, S. 2014. "Multi-Attribute Data Dynamics Discontinuity Identification: An Over-Bounding approach using One-Dimensional Probabilistic Constraints" The Pennsylvania State University College of Engineering Research Symposium, State College, PA, pp. 1-7.
- Laftchiev, E., Lagoa, C., Brennan, S. 2013. "Robust Map Design by Outlier Point Selection for Terrain-Based Vehicle Localization" IEEE Conference on Decision and Control, Florence, Italy, pp. 2822-2827.
- Laftchiev, E., Lagoa, C., Brennan, S. 2013. "Robust Map Design for Terrain-Based Vehicle Localization" The Pennsylvania State University College of Engineering Research Symposium, State College, PA, pp. 1-6.
- Laftchiev, E., Lagoa, C., Brennan, S. 2012. "Terrain-Based Vehicle Localization from Real-Time Data Using Dynamical Models" IEEE Conference on Decision and Control, Maui, HI, pp. 3366-3371.
- Laftchiev, E., Lagoa, C., Brennan, S. 2012. "Terrain-Based Vehicle Localization from Real-Time Data Using Dynamical Models" The Pennsylvania State University College of Engineering Research Symposium, State College, PA, pp. 1-7.

Awards and Fellowships:

- Best Session Presentation American Control Conference 2014.
- Penn State Electrical Engineering Society Graduate Fellowship 2013.
- Best Paper Award - College of Engineering Research Symposium, 2013, 2014.
- Second Place Paper Award - College of Engineering Research Symposium, 2012.
- Nominated for Harold F. Martin Graduate Assistant Outstanding Teaching Award, 2013.

Teaching Courses:

- EE 210: Circuits and Devices - Instructor and TA.
- EE 350: Continuous-Time Linear System - TA